生产企业原材料的订购与运输安排

摘要

针对生产企业原材料采购和运输的规划问题，本文通过数据挖掘，明晰该企业的供应链管理现状，并据此构建供应方的评价体系；而后，采用单目标、多目标、多阶段优化以及现代优化算法，对满足生产、存储、运输等条件的最优方案进行规划，提高了供应链管理效率。

为了建立保障企业生产重要性的数学模型，首先，我们充分探索历史数据，结合文献期刊，从供货能力、速度、稳定性三个维度搭建指标，并对指标进行统计检验和筛选。然后，我们使用 TOPSIS 评价法对 402 家供应商进行评价，确定出 50 家最重要的供应商。

为了预测转运商未来 24 周的损耗率，我们首先对损耗率的周期变化进行分解，接着采用自适应滤波对趋势性进行预测，最后运用时间序列加和模型得到预测结果。而在最少供应商条件下的订购、运输方案的构建过程中，我们采用了多目标贯序算法，首先基于生产、库存需要求解最少供应商为 36 家，再对确定的供应商求解最经济的订购方案，最后采用多阶段优化，对每一周求解损耗最小运输方案。就实施效果而言，订购在保证生产全产能运行的情况下，库存费用仍保持合理较低水平；运输方案的整体损耗率也低于历史均值。

求解材料选取偏好以压缩生产成本的方案时，我们采用了多目标规划，引入松弛变量和柔性约束，对满足生产需求的前提下最大程度压缩成本。本订购方案在低水平的订购成本和仓储成本下，仍将产能浪费率控制在极低水平。而后，在购买方案确定的情况下进行运输方案规划，减少损耗率，最终预计整体损耗率也处于历史低值水平。

为了计算该企业可提高的最大产能，我们从供应商的最大供应量、转运商的最大运输能力以及损耗率入手。扩大选择的供应商范围，综合材料产能以及生产需求，进行最优化求解。

最后，我们评估了我们的模型，通过历史数据验证了方案的合理性。并通过遗传算法对运输问题的求解效率和约束符合度进行了改善。

关键词 供应链管理 TOPSIS 多目标规划 时间序列 遗传算法

1 问题重述

1.1 问题背景

某建筑和装饰板材的生产企业所用原材料总体可分为 A，B，C 三种类型。该企业每年按 48 周安排生产，但是需要提前制定 24 周的原材料订购和转运计划，即根据产能要求确定需要订购的原材料供应商（称为“供应商”）和相应每周的原材料订购数量（称为“订货量”），确定第三方物流公司（称为“转运商”）并委托其将供应商每周的原材料供货数量（称为“供货量”）转运到企业仓库。

已知情况如下：

关于生产。该企业每周的产能为 2.82 万立方米，每立方米产品需消耗 A 类原材料 0.6 立方米，或 B 类原材料 0.66 立方米，或 C 类原材料 0.72 立方米。

关于供货。由于原材料的特殊性，供应商不能保证严格按订货量供货，实际供货量可能多于也可能少于订货量。此外，为了保证正常生产的需要，该企业要尽可能保持不少于满足两周生产需求的原材料库存量，为此该企业对供应商实际提供的原材料总是全部收购。

关于转运。在实际转运过程中，原材料会有一定的损耗（损耗量占供货量的百分比称为“损耗率”），转运商实际运送到企业仓库的原材料数量称为“接收量”。而每家转运商的运输能力为 6000 立方米/周。另外，通常情况下，一家供应商每周供应的原材料尽量由一家转运商运输。

关于费用。实际中 A 类和 B 类原材料的采购单价分别比 C 类原材料高 20%和 10%。三类原材料运输和储存的单位费用相同。

此外，还有该企业近 5 年 402 家原材料供应商的订货量和供货量数据，以及 8 家转运商的运输损耗率数据。

1.2 问题提出

根据以上问题背景，以及所提供的两个附件，建立数学模型解决以下四大问题：

（1）对 402 家供应商的供货特征进行量化分析，建立反映保障企业生产重要性的数学模型，在此基础上确定 50 家最重要的供应商。

（2）为该企业选择最少的供应商满足生产的需求，制定未来 24 周每周最经济的原材料订购方案，并据此制定损耗最少的转运方案。试对订购方案和转运方案的实施效果进行分析。

（3）制定新的订购方案及转运方案，满足尽量多地采购 A 类和尽量少地采购 C 类原材料的前提同时使转运商的转运损耗率尽量少，并分析方案的实施效果。

（4）确定该企业每周的产能可以提高的限度，并给出未来 24 周的订购和转运方案。

2 模型假设

为了简化计算，给出如下合理假设：

（1）供应商的供应能力保持稳定，无明显变化；

（2）在每个生产周期中，如果某转运商在某周的历史转运信息几乎都为没有运送，则认为因为某种原因该转运商在该周无法承担转运任务；

（3）供应商的每周供货量都会在当周转运到企业仓库；

（4）第零周的期末即第一周的期初的库存量恰好满足两周的生产量，即该企业五年前的初始库存量 56400 立方米。

3 变量说明

表 1：变量说明

变量	定义	单位
x_{ik}	第 k 周由第 i 家供应商供应的材料数量	立方米
x_{Ak}	第 k 周 A 类材料的供应量	立方米
x_{Bk}	第 k 周 B 类材料的供应量	立方米
x_{Ck}	第 k 周 C 类材料的供应量	立方米
S_{Ak}	第 k 周 A 类材料的库存量	立方米
S_{Bk}	第 k 周 B 类材料的库存量	立方米
S_{Ck}	第 k 周 C 类材料的库存量	立方米
u_{Ak}	第 k 周 A 类材料的消耗量	立方米
u_{Bk}	第 k 周 B 类材料的消耗量	立方米
u_{Ck}	第 k 周 C 类材料的消耗量	立方米
$plan_{jk}$	第 k 周由第 j 家转运供应的材料数量	立方米
$plan_{i,j,k}$	第 k 周由第 j 家转运第 i 家供应商的材料数量	立方米
$pwhe_{i,j,k}$	第 k 周由第 j 家转运第 i 家供应商的材料数量	立方米
$ploss_{jk}$	第 k 周第 j 家转运商的损失率	
TA_i	第 i 家供应商是否供应 A 类材料的示性函数	
TB_i	第 i 家供应商是否供应 B 类材料的示性函数	
TC_i	第 i 家供应商是否供应 C 类材料的示性函数	
C	技术改造后企业的最大产能	立方米
ξ_k^-	第 k 个柔性约束条件的负偏差量	

ξ_k^+	第 k 个柔性约束条件的正偏差量	
$goal_1$	满足生产约束下的最小转运成本	单位价格
$goal_2$	满足生产约束下的最小存储成本	单位价格
L_i	第 i 周下单次数	次
G_i	第 i 周供货次数	次
M_i	第 i 家供应商未响应的定单数	立方米
P_i	10 个历史周期内第 i 家供货商的平均到货率	立方米
MAX_i	第 i 家企业五年内最大周供应量	立方米
AVE_i	第 i 家企业五年内每个周期最大周供应量的平均值	立方米
MED_i	五年周供应量的中位数	立方米

4 数据探究

《孙子兵法》有云“知己知彼者，百战不殆”，企业管理亦是如此。在着手进行供应链管理的优化之前，我们有必要了解该企业的供应链管理现状，这一方面是为了充分挖掘数据的信息，另一方面更是为了后续进一步优化建模。

经过我们对附件 1 和附件 2 的数据探究，我们发现该企业的供应管理较为粗放，库存管理不甚科学，供应链管理的效率亟待提升。

4.1 订购量和供货量

4.1.1 整体订供量

首先，为了初步了解该企业的供应情况，我们绘制了过去 240 周内的订货量和供货量的折线图。由图 1 我们可以看到，该企业的订货量和供货量波动非常剧烈，但是又有一定的周期性。另外，图中相邻波峰或波谷的间隔大致为 24 周，与该企业 24 周的生产周期相符。

此外，虽然供货量与订货量同波动，但注意到经常有供货量曲线的起伏晚于订货量的现象，说明该企业的订单响应时间不稳定且较长。另外，供货量曲线的下盘整体低于订货量曲线，也说明供货量的不稳且供货不足的情况多发。

为了进一步探究一个生产周期内的采购量和供货量变化特征，我们取五年十个生产周期内每周的平均值，结果如图 2 所示。我们发现，第一周和第五周会出现两个生产周期内的最高峰，这可能是为了预备后期生产所需的原材料的缘故；而之后，每隔三到五周订货量都会出现一个波峰。

另外，供货量在第一、三、五、六周的到货率最高，其他周的供货量都少于订货量，且响应速度更为迟缓。

整体而言，供应的到货率和响应速度较低，供应管理有待加强。

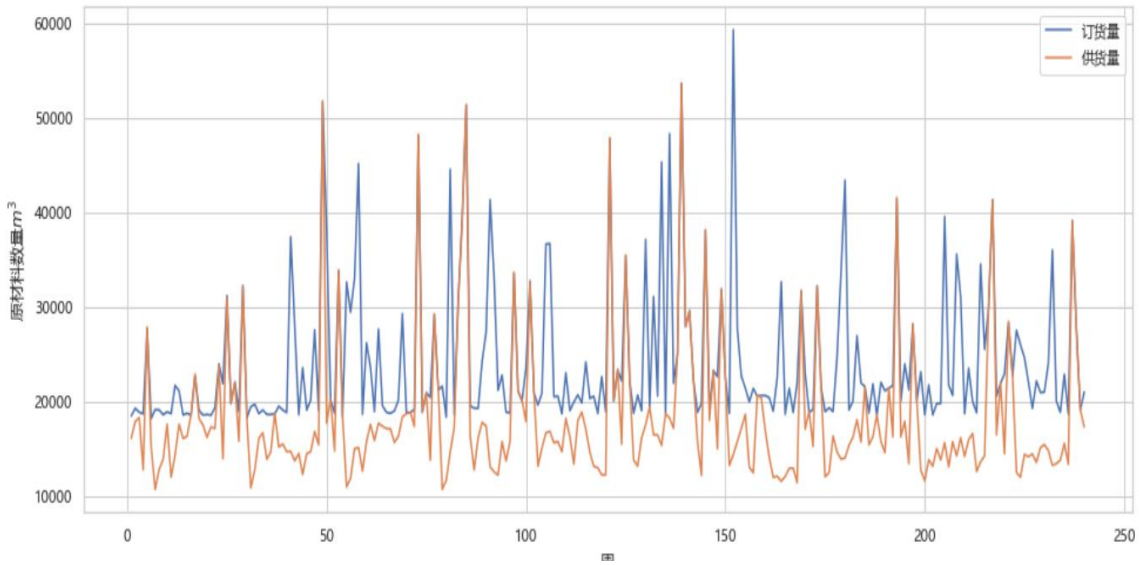

图 1：过去 5 年的订货量和供货量

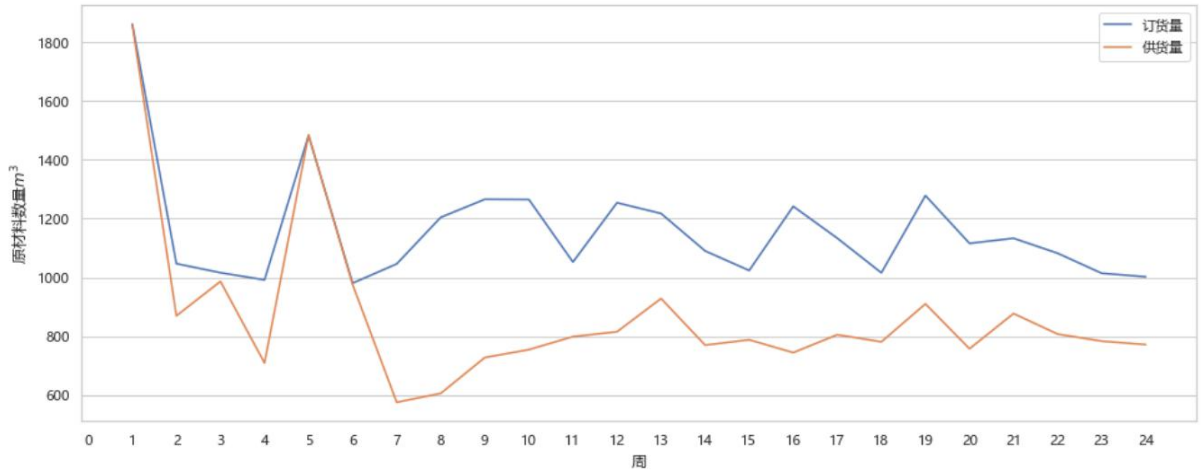

图 2：生产周期内的平均订货量和供货量

4.1.2 订供量的原材料类别差异

考虑到每家供应商都只供应一种原材料，且不同原材料的生产性质和价格也有差异，所以我们进一步从原材料类别上探讨该公司订供量数据的差异。

先看生产周期各类原材料的订供量的变化。就原材料 B 而言，原材料 B 会在生产周期的第一和第五周出现订购和供货的最高峰，联系上文可知，第一周和第五周的订供高峰主要是原材料 B 堆起的。之后原材料 B 的订供量会急速下跌，成为订供量占比最小的原材料，直至最后一周才回升。

而原材料 A 的周均订货量前五周多为 2000 立方米，之后订货量上升到在 3000 立方米上下 1000 立方米徘徊。原材料 C 和 A 类似，订购量在第五周后稳步上升且与原材料 A 不相上下，但是供应量却始终低迷，与 A 存在着 1000-4000 立方米不等的差距。

为了进一步探究各类原材料在订购量与供货量上的不协调性，我们又试着绘制了图 5，展示了生产周期内各类原料平均订供货量差额变化情况。可以看到，只有原材料 A 的订购差额值存在负值，且出现了九周。说明 A 有九周的供应量是超额的，而原材料 B 和 C 的供应量则总是差额，且各周差额整体高于 A。这说明提供原材料 A 的供应商的到货率要整体优于原

材料 B 和 C 的供应商。这也就解释了原材料 A 的订货量不占多数，第五周后的供应量却总是高出 B 和 C 的原因。

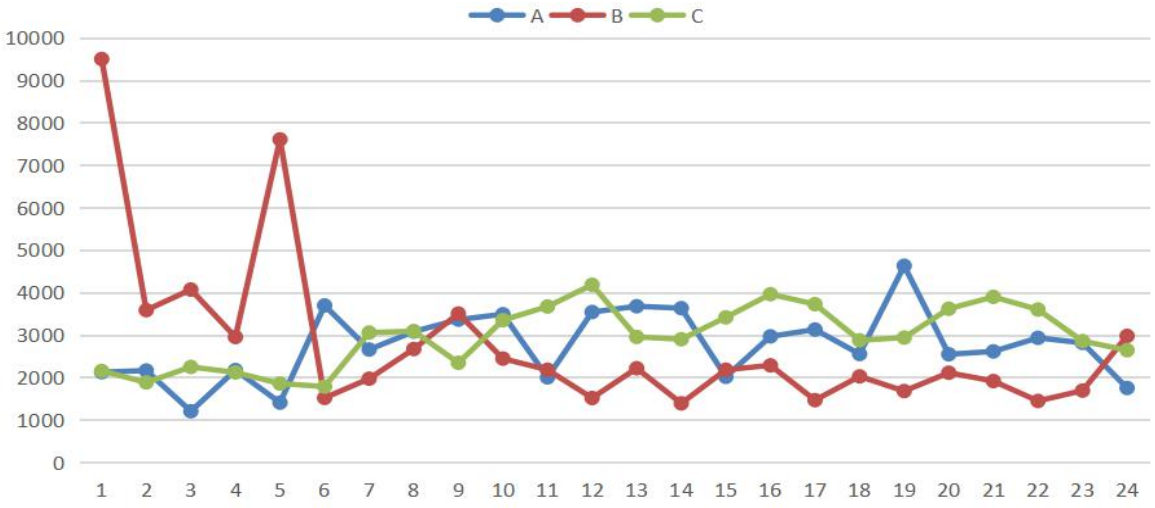

图 3：生产周期内各类原料平均订购量变化

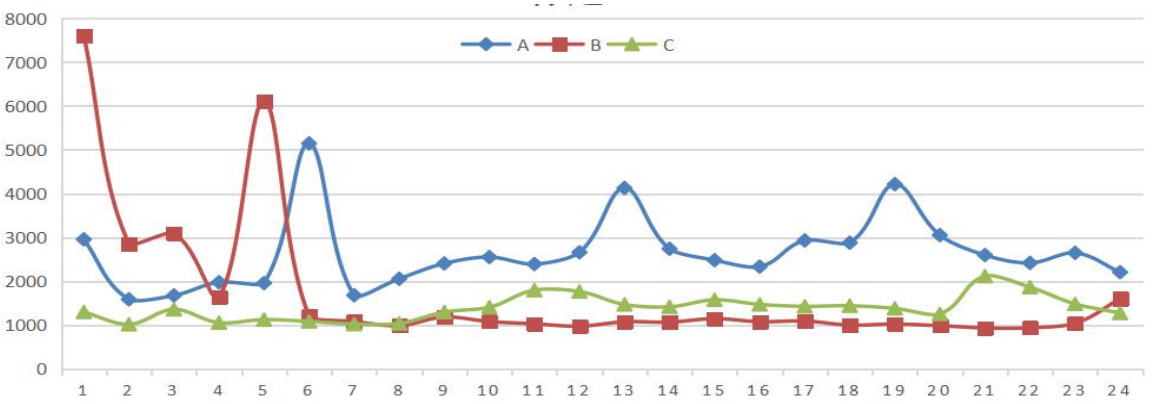

图 4：生产周期内各类原料平均供货量变化

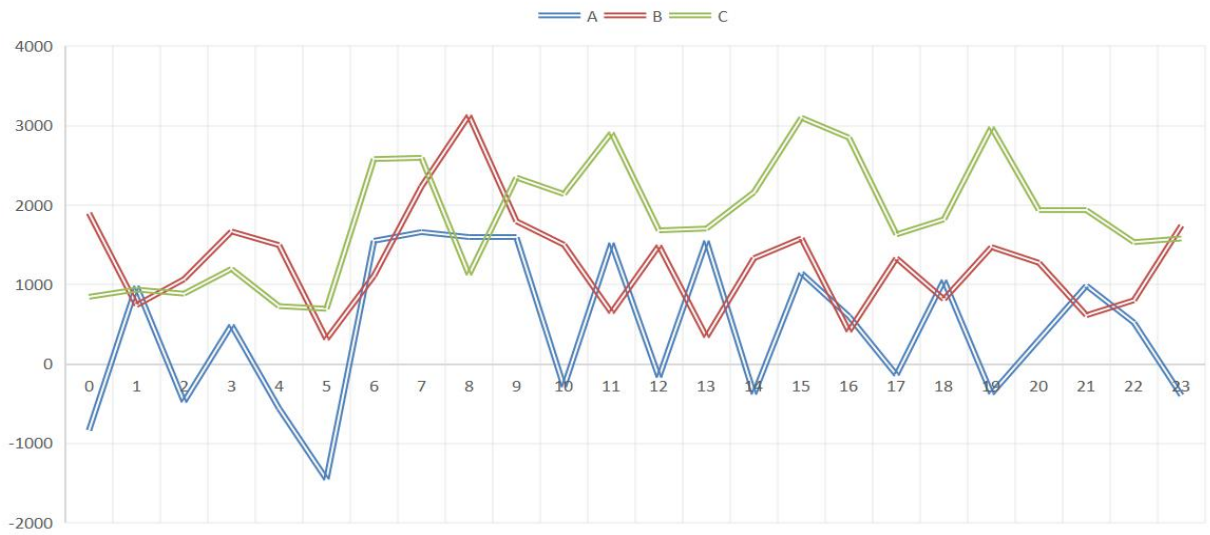

图 5：生产周期内各类原料平均订供货量差额变化

4.2 库存量与损耗率

鉴于原材料性质的差异，且缺乏该企业初始库存中 A、B、C 三类原材料的占比情况，所以为了研究的方便，我们统一以原材料可生产的产品数来衡量库存量。

此外，考虑到实际中该企业的初始库存量不太可能为 0，而为了保证正常生产的需要，该企业又要尽可能保持不少于满足两周生产需求的原材料库存量，所以我们假定第零周的期末即第一周的期初的库存量恰好满足两周的生产量，即该企业五年前的初始库存量 56400 立方米。

另外，“供货量”是指供应商将每周的原材料委托给转运商转运到企业仓库的数量，并不是实际的入库量即“接收量”，还要扣除转运过程的损耗率，才能得到真正的入库量，从而计算库存。所以，我们利用附件 2 的损耗率数据，计算了每周的平均损耗率，并以每周的平均损耗率作为当周的整体损耗率，从而计算每周的转运损耗量，继而以五年总损耗率除以五年总供应量算得五年的总损耗率为 1.249%。观察图 6，我们发现每周的平均损耗率同样有周期性波动的现象，且每周的平均损耗率基本在 0.5%-2.5%之间浮动。

在确定每周的平均损耗率之后，便可以研究该企业过去五年的每周库存和产能利用情况。图 7 展示了该企业以可生产产品数计算的五年历史库存量变化情况。由图我们观察到，前三个生产周期和后两个生产周期都出现了库存为 0 的情况，这表明企业彼时的原材料供应不足以支撑全产能生产，即彼时的原材料供应量不足以生产 2.82 万立方米的产品。

此外，如果以满足企业两周全产能生产的库存量为最优库存量的话，可以发现该企业第四到第八这五个生产周期的库存量多高于最优库存量，特别是第七生产周期，库存一度达到了 20 万立方米，相当于该企业 7 周的产能，库存积压高，造成了沉重的仓储费用；而该企业前三个和后两个生产周期的库存基本都低于最优库存量，说明这五个生产周期内的相对供应短缺，企业生产的稳定性较差，产能制约的风险较大。

当我们进一步探究该企业的产能利用情况时，我们发现该企业有 32 周的产能没有全负荷运转，即全产能运转的比例只有 86.7%，而该企业五年的总产能利用率也只有 98%。通过图 8，我们可以更详细地看到该企业每周的产能利用率。由图可以发现，前三个和后两个生产周期都出现了严重的产能浪费现象，尤其第二个生产周期的产能浪费率一度达到了三分之一。而中间五周虽然一直保持全产能运转，但联系图 7 我们知道，可能是对前期供应量掣肘产能的矫枉过正，彼时的全产能运转却是以大量的积压库存和沉重的仓储费用为代价的，同样并不经济。

综上，我们可以发现该企业的供应商管理较为粗放，库存管理不甚科学，供应链管理的效率亟待提升。

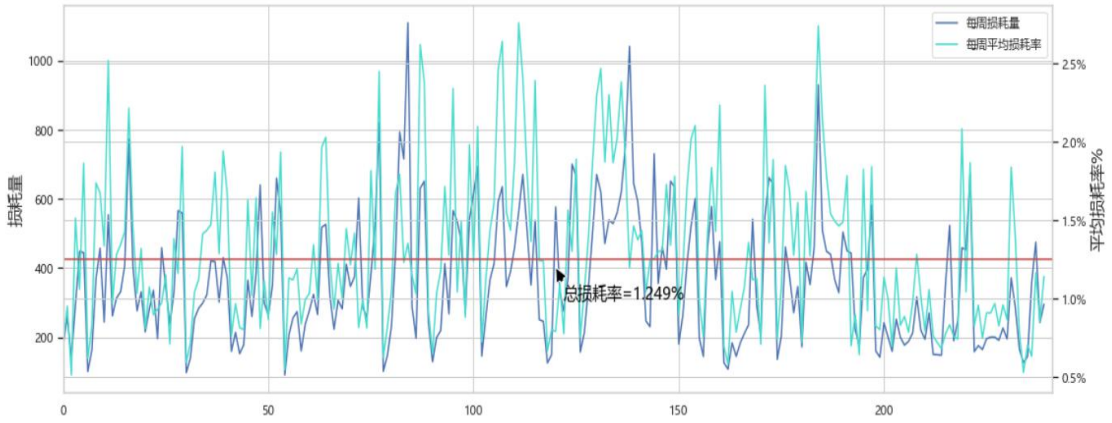

图 6：五年各周损耗量和平均损耗率

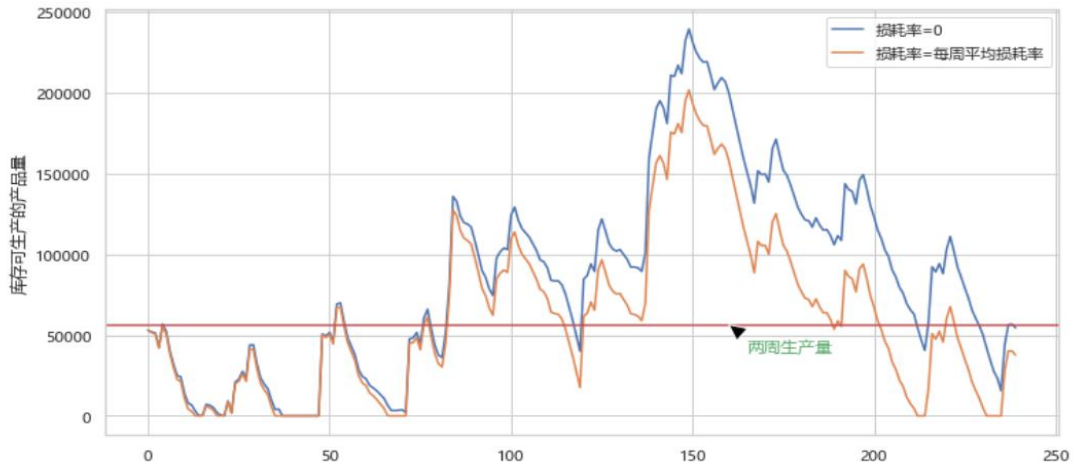

图 7：以可生产产品数计算的五年历史库存量变化

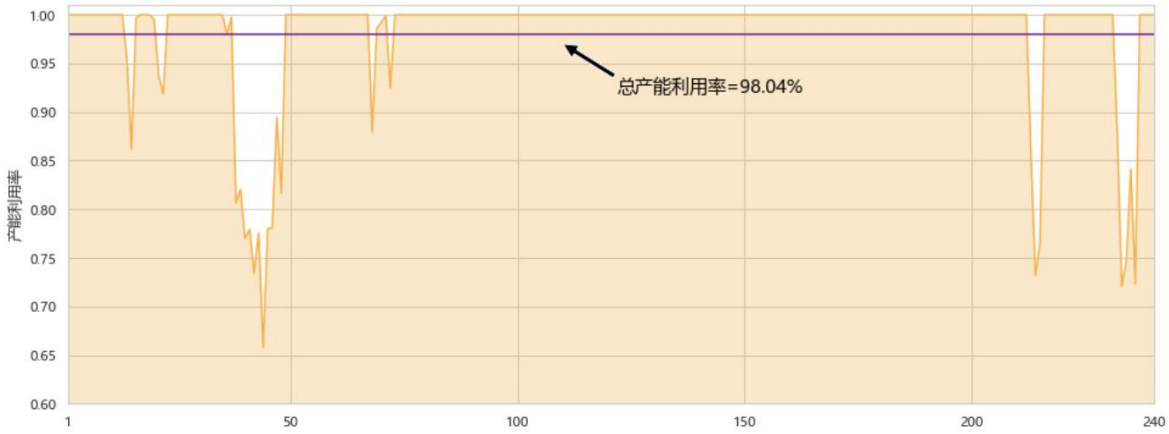

图 8：五年每周的产能利用率

5 模型的建立与求解

5.1 问题一

5.1.1 问题一的分析

问题一的解法主要分三步：第一步，搭建指标体系并进行计算；第二步，指标的检验与筛选；第三步使用 TOPSIS 法型筛选五十家最重要的供应商。

第一步，搭建指标体系并进行计算。为了搭建反映保障企业生产重要性的指标，我们在前期历史数据探索的基础上，结合文献期刊，从供货能力、速度、稳定性三个维度搭建出 12 个指标，分别是 “材料分类”，“下单周次数”，“供货次数”，“订单响应比率”，“平均响应时间”，“加权平均响应时间”，“沉没订单数”，“平均到货率”，“五年最大周供应量”，“五年周供应量的中位数”，“五年生产周期内最大周供应量的平均值”，“SYI”和 “CV”。而后使用 Python 的 pandas 进行 402 家供应商的指标计算。

第二步，对指标进行统计检验和筛选。我们对 12 个原始指标进行 Pearson 相关性分析和多重共线性检验，发现相关性过高的指标有下单周次数，供货次数，订单响应比率，平均到货率，SYI 和 CV，并依据相关性矩阵，决定剔除下单周次数，供货次数，订单响应比率和 CV 四个指标。剔除后多重共线性检验结果表现良好，都在可接受范围内。

最后一步，则是使用 TOPSIS 评价法对 402 家供应商进行评价，为每家供应商打出一个总分，并据此确定出 50 家最重要的供应商。

5.1.2 指标搭建

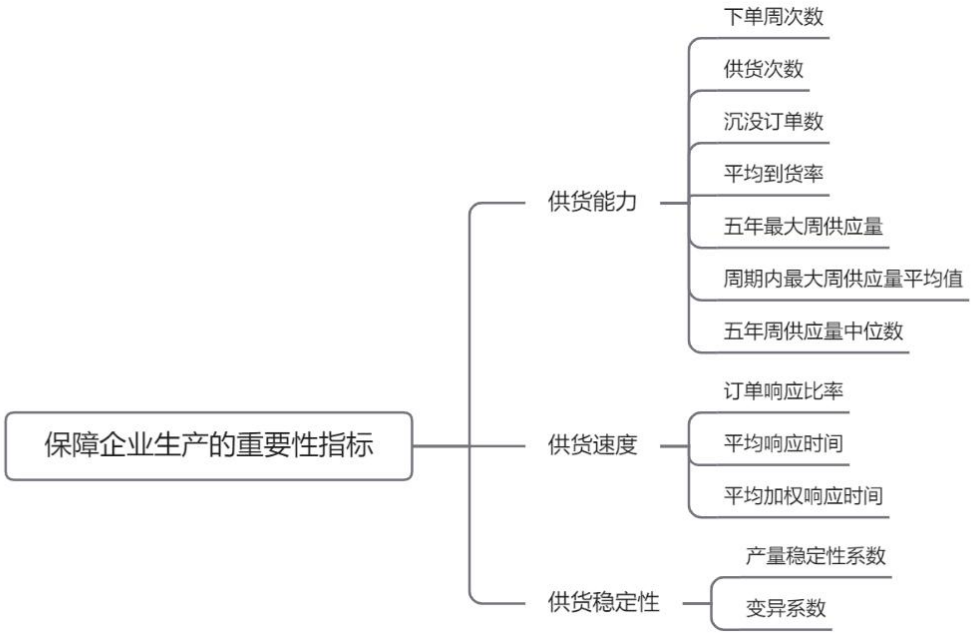

图 9：保障企业生产的重要性指标构建

根据附件 1 数据的特点，为衡量供货商对企业生产的重要性，我们主要从三个方面进行了考虑，包括供货速度、供货能力和供货稳定性。

设 $a_{ij}(i=1,2,\cdots\cdots,402;j=1,2,\cdots\cdots,240)$ 表示对于第 i 家供应商在第 j 周所接到的订货量；$b_{ij}(i=1,2,\cdots\cdots,402;j=1,2,\cdots\cdots,240)$ 表示对于第 i 家供应商在第 j 周的实际供货量。若对于给定的 i 和 j，当 $a_{ij}>0$ 时，记 $ayes_{ij}=1$，否则记 $ayes_{ij}=0$。记总订单量为 L_i，$L_i=\sum_{j=1}^{240}ayes_{ij}$。当 $ayes_{ij}=1$，若 $b_{ij}>0$，则记 $byes_{ij}=1$；若 $b_{ij}=0$，则记 $byes_{ij}=0$。

供货能力方面，我们主要考虑了以下七个指标：

（1） 下单周次数 L_i，$L_i=\sum_{j=1}^{240}ayes_{ij}$。

（2） 供货次数 G_i，$G_i=\sum_{j=1}^{240}byes_{ij}$。

（3） 沉没订单数 M_i，若对于第 i 家供应商在第 j 周，若 $a_{ij}>0$，$b_{ij}=0$，且 $\sum_{j=j}^{j=240}b_{ij}=0$，则记 $m_{ij}=1$，否则 $m_{ij}=0$，则 $M_i=\sum_{j=1}^{240}m_{ij}$。

（4） 平均到货率 P_i，记到货率 p_{iT} 为一个周期（24 周内）的到货率。如从 W1 到 W24 记为周期 1，则 $P_i=\dfrac{\sum_{T=1}^{10}p_{iT}}{10}$。

（5） 五年最大周供应量 MAX_i，为 240 周中的最大的供应量。

（6） 周期内最大周供应量的平均值 AVE_i，记 $\max_{iT}$ 为一个周期（24 周内）的最大周供应量。如从 W1 到 W24 记为周期 1，则 $AVE_i=\dfrac{\sum_{T=1}^{10}\max_{iT}}{10}$。

（7） 五年周供应量的中位数 MED_i，为对第 i 家供应商，240 周中供应量的中位数。

供货速度方面，我们主要考虑了以下三个指标：

（1）订单响应比率 $\alpha_i(i=1,2,\cdots\cdots,402)$：则对第 i 家供应商，$\alpha_i(i=1,2,\cdots\cdots,402)=\dfrac{G_i}{L_i}$。

（2）平均响应时间 $t_i(i=1,2,\cdots\cdots,402)$：对于第 i 家供应商在第 j 周，若 $a_{ij}>0$，$b_{ij}=0$，则记 t_{arrive} 为下一次响应的时间，即表示 $\sum_{j=j}^{j=t_{arrive}-1}b_{ij}=0,\ \sum_{j=j}^{j=t_{arrive}}>0$，则记 $t_{cost}=t_{arrive}-j$，

$$t_i(i=1,2,\cdots\cdots,402)=\frac{\sum_{j=1}^{j=240}t_{cost}}{L_i}$$

（3）平均加权响应时间 $T_i(i=1,2,\cdots\cdots,402)$：$T_i(i=1,2,\cdots\cdots,402)=\dfrac{\sum_{j=1}^{j=240}t_{cost}a_{ij}}{L_i}$

供货稳定性方面，我们主要参考了以下的两个指标：

（1）供货量稳定性指数（SYI_i）：这一指标主要参考了农业中产量稳定性指数 SYI 则 $SYI_i=\dfrac{\overline{b_{ij}}-\sigma_{ij}}{b_{ij\max}}$，其中 $\overline{b_{ij}}$ 表示对第 i 家供应商，所有供应量的平均值，σ 表示标准差，$b_{ij\max}$ 表示供应量的最大值。

（2）变异系数（CV_i）：这一指标主要衡量数据变异程度的大小。$CV_i=\dfrac{\sigma_{ij}}{\overline{b_{ij}}}$

5.1.3 问题一数据处理

首先，利用附件 1 计算指标。我们利用 Python 的 pandas 库，根据 12 个指标的经济学公式，计算的到 402 家供应商的原始指标值。

然后，运用 Python 的 statsmodels 库， 对 12 个原始指标进行 Pearson 相关性分析和多重共线性检验。计算结果如表 2 和图 10 所示：

由表 X 可知，相关性过高的指标有下单周次数，供货次数，订单响应比率，平均到货率，SYI 和 CV。由图 X 可知，这六个指标相关性过高的关系有：下单周次数与供货次数（0.96），订单响应比率与下单周次数(0.75)，订单响应比率与平均到货率(0.88)，SYI 与 CV(-0.86)。所以我们决定剔除下单周次数，供货次数，订单响应比率和 CV 四个指标。剔除后多重共线性检验结果如表 X 所示，剩余指标的方差膨胀因子 (VIF) 和容忍度 (TOL) 都在可接受范围内，即 VIF<10，TOL>0.1。

表 2：多重共线性检验结果

	剔除前		剔除后	
指标	VIF	TOL	VIF	TOL
下单周次数	54.815133	0.01824	-	-
供货次数	43.724642	0.02287	-	-
订单响应比率	45.764331	0.02185	-	-
平均响应时间	3.790391	0.26383	1.7016263	0.58767
加权平均响应时间	1.675608	0.59680	1.4667878	0.68176
沉没订单数	3.998520	0.25009	1.7340318	0.57669
平均到货率	27.349717	0.03656	2.7400314	0.36496
五年最大周供应量	3.147801	0.31768	2.6484072	0.37759
五年周供应量的中位数	1.817618	0.55017	1.5554671	0.64289
五年生产周期内最大周供应量的平均值	2.797422	0.35747	2.7200808	0.36764
SYI	11.036506	0.09061	2.7847883	0.35909
CV	18.31	0.05	-	-

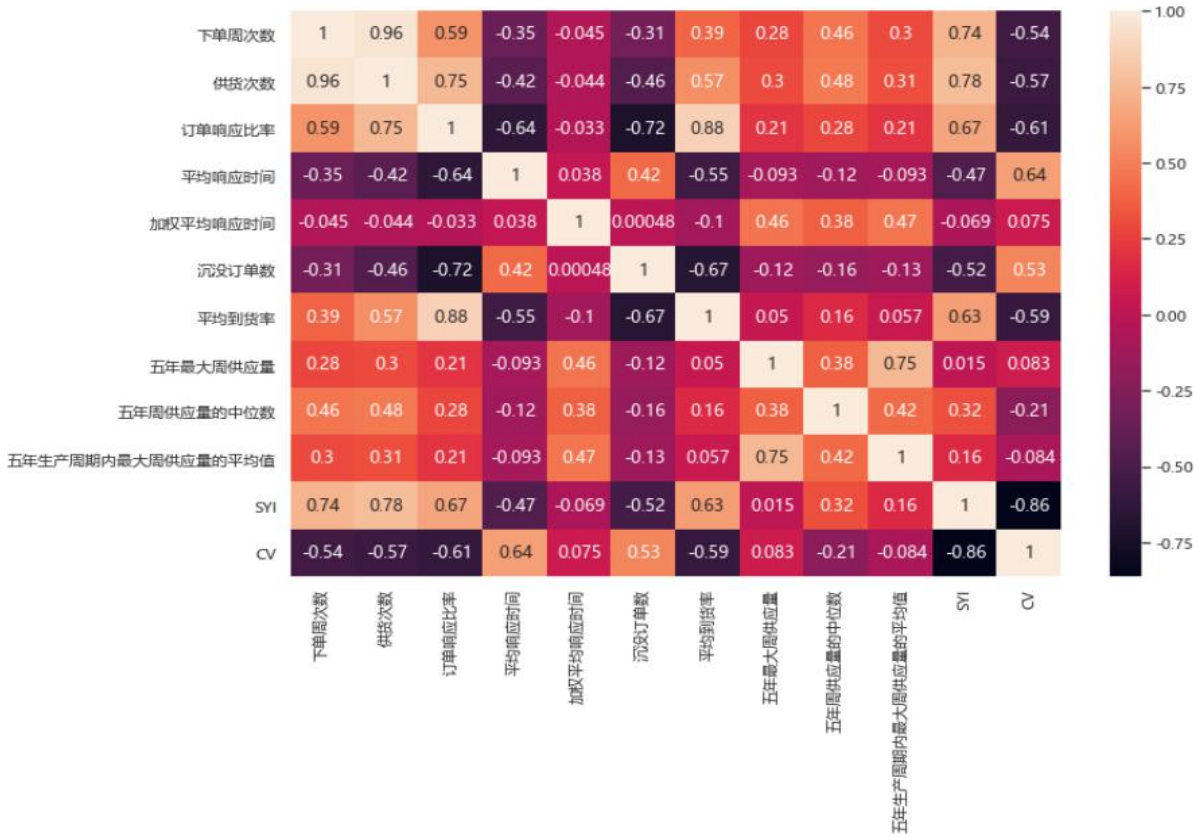

图 10 相关性矩阵

因此，最终的优化指标为：平均响应时间，加权平均响应时间，沉没订单数，平均到货率，五年最大周供应量，五年周供应量的中位数，五年生产周期内最大周供应量的平均和SYI。

5.1.4 TOPSIS 综合评价法筛选优质供应商

TOPSIS 是一种通过借助正负理想解来对各评价问题进行排序的方法，即通过求出各评价对象与正理想解和负理想解的距离，并以此进行排序。

TOPSIS 方法的计算过程如下：对于已有的 402 家供货商和前文已经提取出的 12 个指标：

$$u_{ij}(i=1,2,\cdots\cdots,402;\ j=1,2,\cdots\cdots,12)$$

（1） 对指标进行一致化和无量纲处理，并构造评价矩阵 $V=(v_{ij})_{402*12}$。

（2） 确定正理想解 W^{+} 和负理想解 W^{-}。

$$W^{+}=[w_1^{+},w_2^{+},\cdots\cdots,w_{12}^{+}];\ W^{-}=[w_1^{-},w_2^{-},\cdots\cdots,w_{12}^{-}]。$$

$$w_j^{+}=\max_{1\le i\le 402} v_{ij},\ j=1,2,\cdots\cdots,12;\quad w_j^{-}=\min_{1\le i\le 402} v_{ij},\ j=1,2,\cdots\cdots,12$$

（3） 计算各供应商到正负理想解的距离。

各供应商到正理想解的距离为 $dis_i^{+}=\sqrt{\sum_{j=1}^{12}(v_{ij}-w_j^{+})^2},i=1,2,\cdots\cdots,402.$

各供应商到负理想解的距离为 $dis_i^- = \sqrt{\sum_{j=1}^{12}(v_{ij} - w_j^-)^2}, i = 1,2,\cdots\cdots,402.$

（4） 计算各供应商对理想解的相对接近度 $f_i = \frac{dis_i^-}{dis_i^- + dis_i^+}, i = 1,2,\cdots\cdots,402$

（5） 按 f_i 由大到小排列评价各供应商的优劣次序

通过代码实现，我们得到以下结果:

表 3 TOPSIS 综合评价供应商排名

排名	供应商ID	排名	供应商ID	排名	供应商ID
1	S140	18	S131	35	S86
2	S108	19	S306	36	S346
3	S348	20	S356	37	S294
4	S151	21	S268	38	S80
5	S139	22	S143	39	S244
6	S229	23	S194	40	S218
7	S361	24	S352	41	S114
8	S307	25	S37	42	S7
9	S374	26	S338	43	S292
10	S330	27	S284	44	S266
11	S308	28	S247	45	S123
12	S395	29	S55	46	S291
13	S282	30	S365	47	S76
14	S340	31	S31	48	S221
15	S126	32	S364	49	S67
16	S275	33	S40	50	S169
17	S329	34	S367		

5.2 问题二

5.2.1 原材料订购方案

在满足生产需求的前提下，我们首先根据题目要求计算了最少的供应商选择，接着在前最少供应商的基础下，求解最经济的订购方案。我们使用 lingo 多目标贯序算法进行计算，以满足生产需求为第一优先级，最少供应商为第二优先级，最经济为满足以上条件的第三优先级。

针对目标一，企业的生产约束主要是以下几个方面：

（1）企业每周的产能为 28200 立方米，因此对每周使用的 A,B,C 类原材料的使用量 u_{Ak}, u_{Bk}, u_{Ck} $k = 1,2,...24$，有约束：

$$\frac{u_{Ak}}{0.6} + \frac{u_{Bk}}{0.66} + \frac{u_{Ck}}{0.72} = 28200$$

（2）企业尽可能保持不少于满足两周生产需求地原材料库存量，因此对每周库存 A,B,C 类原材料 S_{Ak}, S_{Bk}, S_{Ck} $k = 1,2,...24$，有约束：

$$\frac{S_{Ak}}{0.6}+\frac{S_{Bk}}{0.66}+\frac{S_{Ck}}{0.72}\geq 28200*2$$

由于每周库存是尽可能不少于两周生产需求，因此我们认为该条件是柔性约束，引进松弛变量将不等条件变为等式条件，从而能在更大的库存范围求解，对于初始库存为 0 的前两周起到了缓冲效果，即

$$\frac{S_{Ak}}{0.6}+\frac{S_{Bk}}{0.66}+\frac{S_{Ck}}{0.72}+\xi_k^- -\xi_k^+ = 28200*2$$

（3）企业的各个供应商有供应能力的限制，因此对第 i 家供应商在第 k 周的材料供应量 x_{ik} 有约束：

$$x_{ik}\leq \max_k x_{ik}$$

这里我们采用第 i 家供应商各年度的最大周供应量的平均值作为该供应商的最大供应量。

（4）对企业生产的递推数量变化，对于每一类材料，第 k 期库存量等于第 k-1 期库存加上第 k 期材料供应量减去第 k 期材料使用量，因此有约束：

$$S_{Ak}=S_{A,k-1}+x_{Ak}-u_{Ak}$$
$$S_{Bk}=S_{B,k-1}+x_{Bk}-u_{Bk}$$
$$S_{Ck}=S_{C,k-1}+x_{Ck}-u_{Ck}$$

（5）各类材料的供应量等于该类供应商的供应量之和，因此有

$$x_{Ak}=\sum_i x_{ik}TA_i$$

其中 TA_i 是第 i 家供应商是否供应 A 类材料的示性函数，即

$$TA_i=\begin{cases}1 & \text{第i家企业供应A类材料}\\ 0 & \text{第i家企业不供应A类材料}\end{cases}$$

同理，对 B,C 类材料的供应量和示性函数 TB_i, TC_i 也有类似定义。

目标一为满足生产需要，也就是在刚性约束(1)(3)(4)(5)下，求柔性约束的表示负偏差的松弛变量的 ξ_k^- 尽量小，即

$$\min \quad z_1=\sum_k \xi_k^-$$

因此，具体的求解步骤为：

（1）求解目标一的最小值，目标值为 5338.15，根据贯序算法，限定优先级为一的目标值 $z_1=5338.147$ ，对最小供应商数目进行求解，通过 lingo 求得最小供应商数为 36。

（2）取 $z_1=5338.147, z_2=36$, 求解最小的成本。这里的成本包括材料价格、运输费用和储存费用，考虑到每立方米的 A,B,C 类材料的价格比例为 120%:110%:100%，运输费用和储存费用的单价相同，我们设第三优先级下的目标函数为：

$$\min z_3 = \sum_k (1.2x_{Ak} + 1.1x_{Bk} + x_{Ck}) + 2\sum_k (S_{Ak} + S_{Bk} + S_{Ck})$$

从而使用 lingo 求得在最小费用下的购买方案。

5.2.2 转运方案

这一部分我们主要我们利用 *Python* 的 *statsmodels* 库的 *seasonal_decompose* 函数，对八家转运商的历史转运损耗进行 *decompose* 数据分解，来求转运商损耗率。

Decompose 数据分解模型主要有两类：相加模型（*additive*）和相乘模型（*multiplicative*）。

相加模型：$Y(t) = T(t) + S(t) + e(t)$

相乘模型：$Y(t) = T(t) * S(t) * e(t)$

其中，$T(t)$是趋势项，$S(t)$是季节性周期项，$e(t)$是残值项。一般地，理想的分解模型中残值项应该是一个均值为 0 的随机变量。而我们主要采用的是相加模型，下面对相加模型的思路做简单的介绍。

首先，分解趋势项，采用中心化移动均值的方法。

$$T_t = \frac{x_{t-(\frac{f-1}{2})} + x_{t-(\frac{f-1}{2})+1} + \cdots\cdots + x_{t+(\frac{f-1}{2})}}{f}, t \in (\frac{f+1}{2}, l - \frac{f-1}{2})$$

当f为奇数时采用上述计算方法。

$$T_t = \frac{0.5x_{t-(\frac{f}{2})} + x_{t-(\frac{f}{2})+1} + \cdots\cdots + x_{t+(\frac{f}{2})-1} + 0.5x_{t+(\frac{f}{2})}}{f}, t \in (\frac{f}{2}+1, l - \frac{f}{2})$$

当f为偶数时采用上述计算方法，其中T_t是趋势项，f是时间序列频率，l是时间序列长度。结果为长度为 l 的时间序列。为便于后续的向量计算，当 t 超出上述下标的定义域时，其值为 *NA*，如T_1。

第二步，分解季节性周期项。

A.采用将原始时间序列减去趋势项。$S_t = x_t - T_t$

B.将各个周期内相同频率下的值平均化，得到季节项 *figure*

$$figure_t = \sum_{i=0}^{n} \frac{S_{t+i*f}}{f}, t \in (1, f), n = l\text{对}f\text{取整}, \text{即}\max(n, nf \le l)$$

C.将 *figure* 中心化，得到中心化的季节项 *Figure*，代码可表述为

$$figure = figure - mean(figure)$$

最终得到了长度为 f 的季节项。

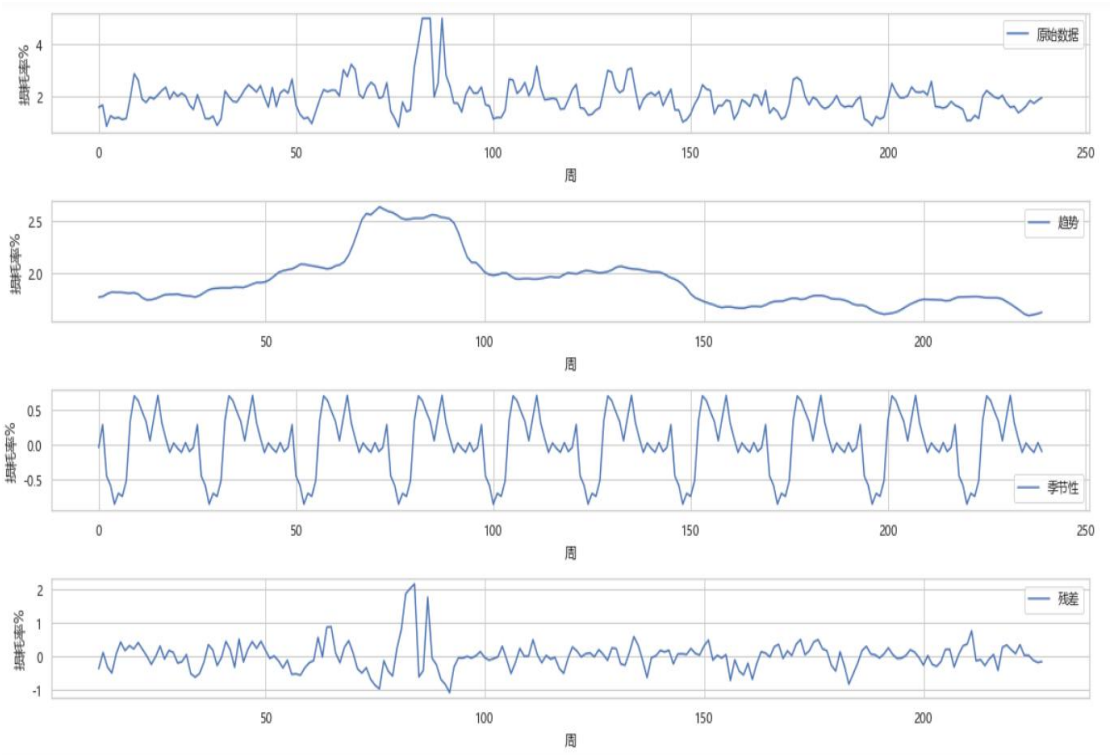

图 11 转运商 T1 的数据分解

我们首先对每一家转运公司的损耗率的五年变化绘制折线图，发现同一家转运公司的损耗率具有很强的周期性。因此我们通过相加模型把他们分解为趋势项、周期项和随机扰动项。

考虑到趋势曲线本身的趋势性，我们通过自适应滤波法对趋势项进行预测。不同于移动平均法，自适应滤波法是通过用一组给定的权数来计算预测值，然后计算预测误差，在根据误差来调整权数以减少误差，如此反复进行直到使误差减少到低程度的预测方法。由于随机扰动项的期望是 0，因此预测的值是周期项和趋势项的和。

在得到了未来 24 周各个转运商的损耗率预测后，我们对最小损耗的转运方案进行规划，每一周运输的材料即为该周供应商供给的材料，因此我们可以将运转分为不同的阶段，每一个阶段为一周，实现多阶段规划。由于整体最优化的子过程也必然是该过程的最优化，又根据假设当周供应当周运输，因此我们对每一周进行最小损耗规划，得到的结果的解集就是 24 周最小损耗的最优解。

对固定的周次 k=1,2…24，是否由第 j 家转运商运输第 i 家供应商的材料的 0-1 变量 $pwhe_{i,j,k}$ 以及由第 j 家转运商运输第 i 家供应商的量 $plan_{i,j,k}$

目标函数为

$$\min \quad z=\sum_{i}\sum_{j} plan_{i,j,k}\, ploss_{j,k}$$

其中 $ploss_{i,j,k}$ 是第 j 家转运商在第 k 周的损耗率预测

有如下约束条件：

(1) 各企业的该周供应量被全部运输

$$\sum_{j} plan_{i,j,k}=x_{i,k}$$

(2) 各转运商一周的转运量最大为 6000 立方米

$$\sum_i plan_{i,j,k} \le 6000$$

(3) 一家供应商每周供应的原材料尽量由一家转运商运输

$$\sum_j pwhe_{i,j,k} + \xi_i^- - \xi_i^+ = 1$$，其中 ξ_i^-, ξ_i^+ 分别是正、负偏差松弛变量

通过 lingo 进行多阶段规划求解得到损失率最小的转运方案。

5.2.3 方案效果评估

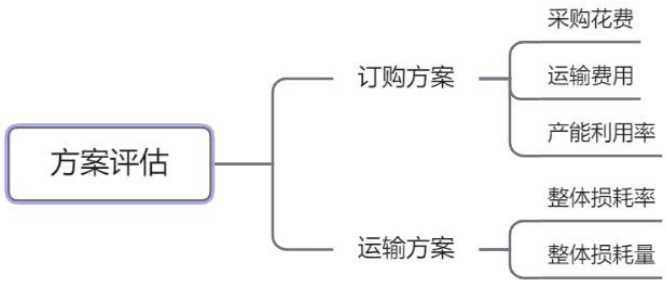

图 12 方案效果评估

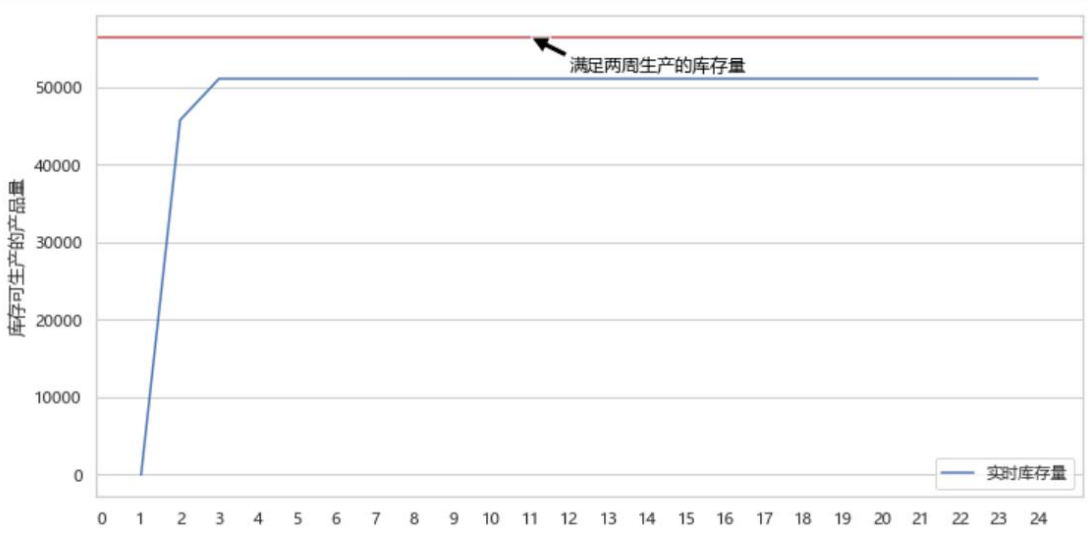

图 13 方案预计库存可生产产品量

对于订购方案，我们的订购总量为 457525.6 立方米，以原材料 ABC 分别记作 1.2，1.1 和 1 的价格下的采购花费为 535356.0052674994，产能利用率为 100%。

从采购花费来看，我们参考了初始库存量类似的第四和第六生产周期，第 4 生产周期的采购花费 540766.2，第六生产周期的采购花费 590277.8。可以看出，我们的订购花费更低。同时，从产能利用率和库存管理上，我们的订购方案也克服了之前产能利用率和库存管理难以兼顾的问题。从产能利用率上看，我们方案的产能利用率为 100%。从库存管理上看，由上图可以看出，库存可生产的产品量基本满足两周生产。

对于运输方案，我们的运输费用为 457525.6362243776，运输的整体损耗率为 1.202%，也得到了较好的效果。

综上所述，我们的方案在产能利用率和库存管理上取得了一定平衡，也实现了费用的节约。

5.3 问题 3 的求解

企业为减少转运以及仓储的成本，考虑到单位体积 A 材料产生的产能最大，单位体积 C 材料产能最小，希望尽可能多地采购 A 类材料，尽可能少地采购 C 类材料。与此同时，企业希望尽量减少运输损耗。

我们将问题三的求解分为购买和运输两个部分，对两个部分分别采用多目标优化进行求解。

5.3.1 采购方案

在购买阶段，第 k 周由第 i 家供应商的供应量 $x_{i,k}$ 满足以下条件：

(1)满足生产需求产能消耗和储存约束，与这里的处理与第二问相同，即

$$\frac{u_{Ak}}{0.6}+\frac{u_{Bk}}{0.66}+\frac{u_{Ck}}{0.72}=28200$$

$$\frac{S_{Ak}}{0.6}+\frac{S_{Bk}}{0.66}+\frac{S_{Ck}}{0.72}+\xi_k^- -\xi_k^+ =28200*2$$

(2)尽可能多的采购 A，尽可能少的采购 C:

$$x_{Ak}+\xi_{k+24}{}^- -\xi_{k+24}{}^+ =\max X_{Ak}$$

$$x_{ck}+\xi_{k+48}{}^- -\xi_{k+48}{}^+ =0$$

这里的 $\max X_{Aj}$ 取一周产能全部由 A 类材料产生的情况下的使用量。

(3)转运和存储成本尽量小：

转运成本：

$$\sum_k (x_{Ak}+x_{Bk}+x_{Ck})+\xi_{73}{}^- -\xi_{73}{}^+ =goal_1$$

存储成本：

$$\sum_k (1.2x_{Ak}+1.1x_{Bj}+x_{Cj})+\xi_{74}{}^- -\xi_{74}{}^+ =goal_2$$

其中 $goal_1$，$goal_2$ 分别是满足生产需求约束条件下的最小转运成本和运输成本分别为 442676.8 和 779384.8 个单位价格。

目标函数为：

$$\min z_1=\sum_{i=1}^{24}\xi_i^+ +\sum_{i=48}^{74}\xi_i^+ +\sum_{i=24}^{48}\xi_i^-$$

5.3.2 运输方案

在得到满足以上条件的最优购买方案后，我们进行最小损耗运输方案的规划。

运输方案应该满足的条件有：

（1）一周内一家转运商的运输总量不能超过 6000 立方米，即

$$\text{对每一个}k\text{和j}:\sum_i plan_{i,j,k}\le 6000$$

（2）目标函数为运输损耗最小：

$$\min z_2 = \sum_i \sum_j plan_{i,j,k} ploss_{j,k}$$

使用 lingo 求解得到损耗最小的运输方案。

5.3.3 方案效果分析

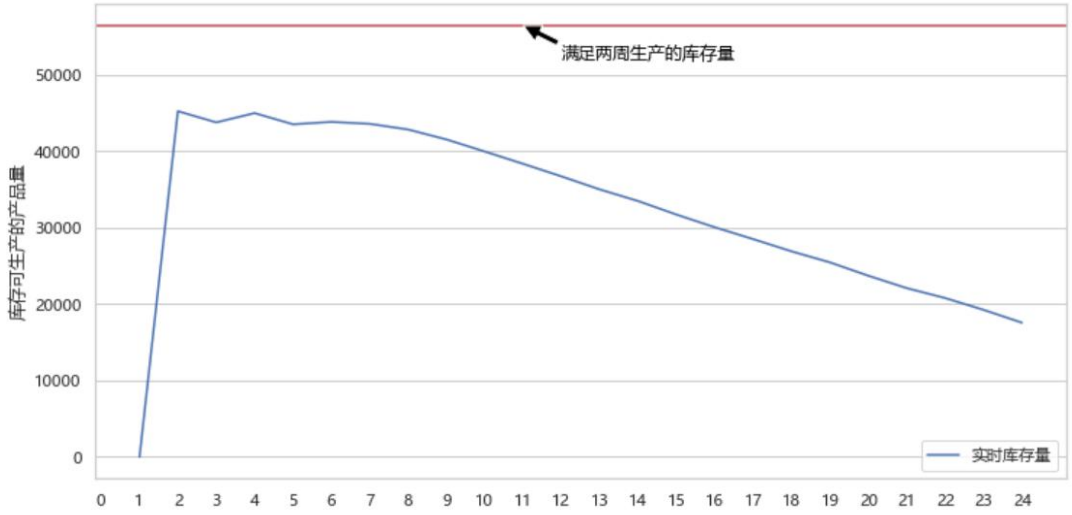

图 14 方案预计库存可生产产品量

对于采购方案，以原材料 ABC 分别记作 1.2，1.1 和 1 的价格下，采购花费为 511661.，总订购量为 442652 立方米。库存则保持在 1 到 2 周生产量的低位，仓储费用大为减少。即使采购和仓储费用维持在低水平，但产能利用率除了第一周有 214.497 立方米的产能缺口外，其余隔周都保持着全产能运转。

而在此转运方案下，总损耗为 5207 立方米，整体损耗率为 1.1763%，较 1.249%的五年平均损耗率改观良多。

5.4 问题 4 的求解

企业能够提高的最大产能主要取决于企业自身、供应商和运输链，企业能够通过技术改进提高产能，因此问题四主要关注供应商和转运商对企业最大产能的约束。供应商和转运商是企业生产材料的来源，供应商对材料的限制主要是对材料的供应能力，我们结合供应商本身的信誉、响应发货时间等指标，对供应商进行筛选，决定以问题一排名在前 180 的供应商为供应端。转运商对材料的限制分为两个方面，一是转运商的最大运输量，二是转运商的损耗率。因此我们对企业的最大产能 C 得出以下约束条件：

(1) 企业能够满足维持生产的基本要求：

$$\frac{u_{Ak}}{0.6} + \frac{u_{Bk}}{0.66} + \frac{u_{Ck}}{0.72} = \mathrm{C}$$

$$\frac{S_{Ak}}{0.6} + \frac{S_{Bk}}{0.66} + \frac{S_{Ck}}{0.72} + \xi_k^- - \xi_k^+ = 2C$$

$$S_{Ak} = S_{A,k-1} + x_{Ak} - u_{Ak}$$

$$S_{Bk} = S_{B,k-1} + x_{Bk} - u_{Bk}$$

$$S_{Ck} = S_{C,k-1} + x_{Ck} - u_{Ck}$$

$$x_{Ak} = \sum_i x_{ik} TA_i$$

$$x_{\mathrm{B}k} = \sum_i x_{ik} TB_i$$

$$x_{Ck} = \sum_i x_{ik} TC_i$$

(2) 企业接收的材料量受到供应商最大供给量的限制：

$$x_{ik} \le \max_k x_{ik}$$

(3) 一家转运商一周的最大运输量是 6000，供应的材料不能超过最大运输量：

$$\sum_i plan_{i,j,k} \le 6000$$

$\sum_i x_{i,k} \le \max T$ ，maxT 为最大接收量，不同于转运量

考虑到各个转运商在各周不同的损耗率预测值，对接收的材料的约束为：

$$\sum_i x_{i,k} \le \sum_j 6000 * ploss_{jk}$$

进一步讨论，为了求解出最大的产能，我们发现 A 类材料的单位产能大于 C 类材料，因此想要产能最大化，应该购买尽量多的 A 类材料和尽量少的 C 类材料，从而产生针对材料种类的柔性约束。另外，由于供应商具有不同的供应材料类别以及最大供应的能力等指标，供应商也会相互竞争损耗率较小的转运商，因此最大产能也和选择的供应商有很大关联。

综合考虑以上原因，同时也为了减少计算的复杂度。我们对不同的供应商选择进行了计算最大产能。从原本的 50 家企业开始，我们取 5 为间隔，计算了不同企业数目的最大供应能力，发现在 115 家企业处离散取值最大。接着我们在 115 家企业附近进行计算，得到选择前 119 家企业时得到的产能最大，如图：

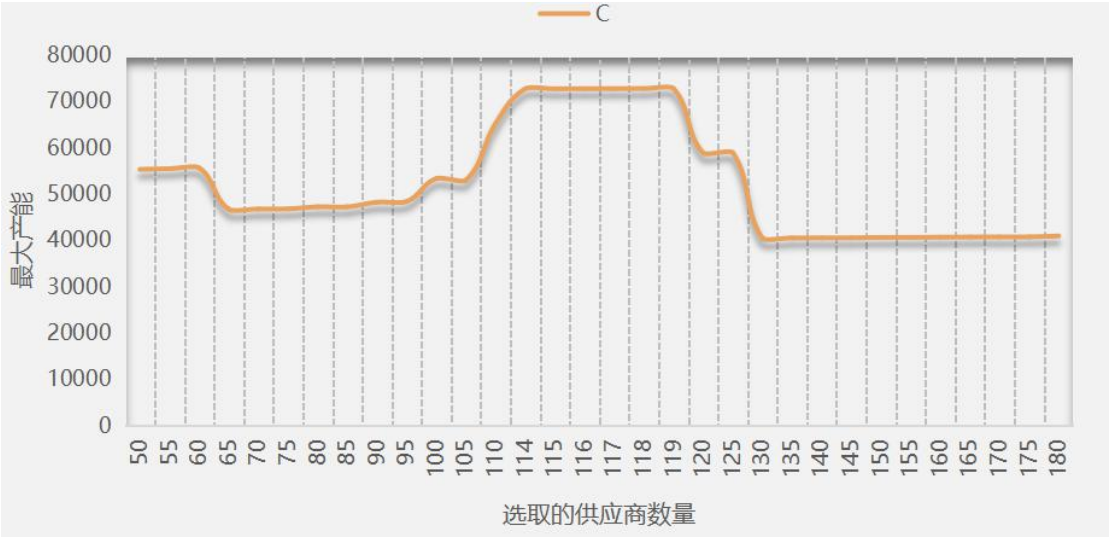

图 15 方案预计选择供应商数量与最大产能

我们对 119 家企业进行了多目标规划，通过 lingo 求解规划结果。

6 模型优缺点

6.1 模型优点

（1） 评价模型客观全面，供货特征量化科学。我们参考了权威期刊和论文，细致探讨供应商指标，通过相关、共线性检验对指标进行评估和筛选。

（2） 损耗率估计可靠，采用加法原理分解具有周期性的时间序列。

（3） 结合了多种优化方法构建优化模型，结合单目标、多目标、多阶段优化，引进松弛变量，对模型范围进行适当的扩大和调整。

6.2 模型缺点

（1） 模型对初始库存取值为 0，事实上企业的库存在期初不一定为 0，需要进一步的泛化和讨论。

（2） 模型未精确讨论供应商发货的延时效应，由于绝大多数排名前五十的供应方的平均发货等待周数小于 0.5，假设供应方当周发货，事实上存在延时效应，可以进一步调整方案。

6.3 模型改进

6.3.1 对最大供应量的调整

我们一开始对供给商的最大供应量时各年最大周供应量的平均值。但是从供给商五年的供给数据可以看出，供给商的供应量时一个随机变量，通过直方图可以大致看出供给商的供给量分布。从实际出发，供给商的供应能力受到自然资源、自身供应链条的影响，很难达到最大供应量。因此我们对次做出调整，我们结合平均最大供应量和供应量的中位数，考虑一年内的波动，引进随机数，将改进后的最大供给量表示为：

$$\max_k x_{ik} = r_k AVE_i + (1-r_k)(AVE_i + MED_i)$$

$avemaX_i$ 表示第 i 家供应商五年平均周供应量最大值，$midX_i$ 时该供应商五年内的供应量中位数。我们将最大供应量的调整运用于第 2，3，4 题的计算，效果良好。

6.3.2 遗传算法对运输指派的改进

由于第三题和第四题的运输分配问题变量维度较高，并且引进了较多 0-1 变量，使得 lingo 计算的时间复杂度较高，对一家供应商尽量由一家转运商运输的条件较难满足。基于问题的复杂性和采用多项式解法的困难性，我们用遗传算法对运输分配问题进行了改进。

遗传算法是一种模拟遗传选择和自然淘汰的计算模型，本质是通过群体搜索技术和逐代进化，最终得到最优解。遗传算法的步骤主要包括：编码、初始化、适应度筛选、交叉、变异，进行循环更新。

在成本最小的确定性购买方案下考虑运输指派问题，由于第 k 周从第 i 家企业的购买量已固定，我们将这个购买量作为第 k 期遗传算法种群的上边界，将 0 作为下边界，采取实数编码，用随机数的配比表示购买量的配比，实现可行域的确定。

另外，我们的适应度主要争对损耗量，结合超过一家转运商运输的供应商的个数，进行计算。此外，我们还对单个转运商的总运输量超过 6000 立方米的情况设置罚函数项，实现越界个体的筛除。

最后，考虑到一周选中的供应商具有一定的稀疏性，我们对交叉的位置进行了非空的筛选。为了限制供应商的转运商数量，我们对转运商随机数编码进行了适当的稀疏化。

改进后的遗传算法对转运商数量限制效果有明显改善，同时具有良好的收敛特性，例如对第三题的转运方案第七周的适应度函数代际变化如下：

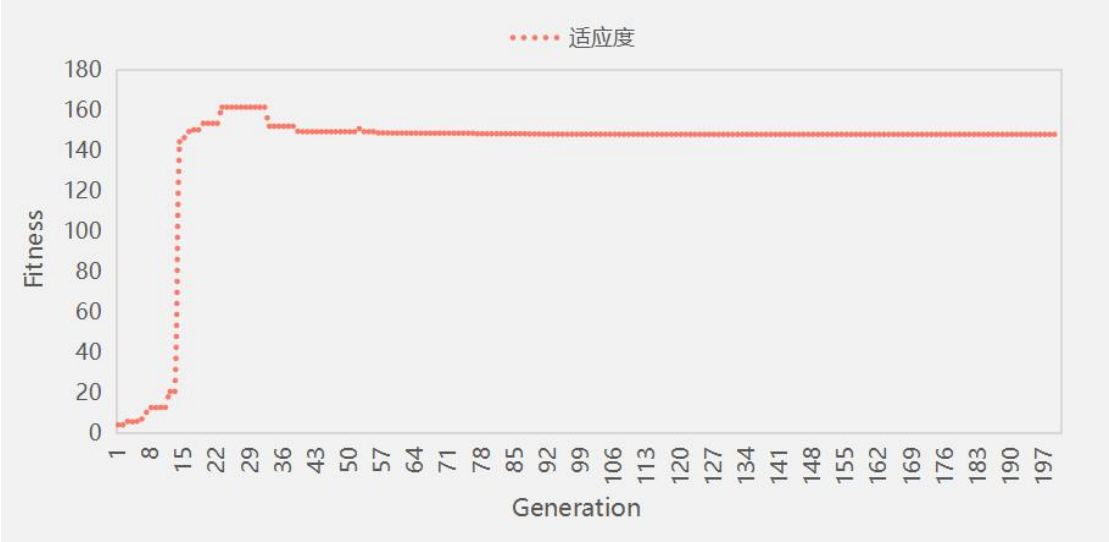

图 16 适应度函数代际变化

由于参赛时间的限制，我们并未对第三、四题的每一个阶段都进行遗传算法的改良，如果有更加充足的时间，我们将能够对现有方案做进一步优化。

7 参考文献

[1]李肇坤,郭贝贝,杨赞.港口物流服务供应链中多任务分配问题研究[J].水运工程,2009(05):39-43.
[2]高翔宇.基于目标导向的供应商业绩评价办法[J].电子质量,2021(08):82-86.
[3]李英,耿玉德.木材加工企业供应链管理研究[J].中国林业经济,2006(05):25-28.
[4]王喆,邵鸿远,丛子皓,马雯雯.考虑供应商聚类的应急医疗物资协同配送仿真[J/OL].系统仿真学报:1-10[2021-09-12].
[5]马力,杨林章,沈明星,夏立忠,李运东,刘国华,殷士学.基于长期定位试验的典型稻麦轮作区作物产量稳定性研究[J].农业工程学报,2011,27(04):117-124.
[6]王捷捷.浅谈生产制造企业中采购风险的控制[J].中小企业管理与科技(中旬刊),2021(10):49-51.

附录：

本论文使用了 Python、Matlab 与 lingo 代码。具体代码如下。

TOPSIS 综合评价法

```
score=xlsread('zhibiao2.xlsx','C2:J403');
maxsco=max(score);
minsco=min(score);
%归一化
%正向的稳定、供货能力、供货率
%反向相应时间
pos=[4,5,6,7,8];
neg=[1,2,3];%反向列
score(:,pos)=(score(:,pos)-minsco(pos))./(maxsco(pos)-minsco(pos));
score(:,neg)=(max(neg)-score(:,neg))./(maxsco(neg)-minsco(neg));
for i=1:size(score,2)
    score(:,i)=score(:,i)./sqrt(sum(score(:,i).*score(:,i)));%正向化矩阵标
准化
end
smax=max(score);
smin=min(score);

for j=1:size(score,1)
    distp(j)=sqrt(sum((smax-score(j,:)).^2));%D+
    distm(j)=sqrt(sum(score(j,:)-smin).^2);%D-
end
newscore=distm./(distp+distm);
D=[distp' distm'];
[sd,dex50]=sort(newscore,'descend');
ans=[sd' dex50'];
original=xlsread('zhibiao2.xlsx','C2:J403');
final50=original(dex50,:);%提取前50各项指标
```

自适应滤波

```
function w=waveadj(y,k,N)%自适应滤波算法
T=length(y);
w=1/N*ones(1,N);%初始化权重
wlast=ones(1,N);%初始化上一步权重

 while sum(abs(w-wlast))>0.01 %当和上一步权重差距仍然较大时，再来一次

        %一轮调整权重
        for i=N+1:T
            yhat=sum(w.*y(i-N:i-1));%预测值
            err=y(i)-yhat;%误差计算
```

```
            wlast=w;
            w=w+2*k*err*y(i-N:i-1);%更新权重，用w'=w+2ke(t+1)y(t-i+1)
            w=w/sum(w);%归一化
        end

 end
end
```

贯序算法

```
model:
!the multigoal;
!1st min factory;
!2nd min price;
sets:
  amount/1..24/:uA,uB,uC,xA,xB,xC,SA,SB,SC,rdm;
  !used amount in each month,the amount to buy in each amount,the amount
of store in each month;
  buy/1..50/:f,maxp,mid,TA,TB,TC;!the factories,the max production of
factories,which type to buy,the rand use to adjust the max;
  week/1..24/:w;!which week;
  res(buy,week):mb,ans;!the amount to buy in the factory;
  goal/1,2/:z,g;!the different goals,the best of z;
  tylimit/1,2,3/:limit;!the limit in amount of each type;
  soft_variable/1..24/:dm,dp;!soft conditions;
endsets

data:
  seed=123;
  rdm=@qrand(seed);!use rand to change the max;
  mid=@ole('D:\MCM2021\fact50.xlsx','mid');!the midiam;
  TA=@ole('D:\MCM2021\fact50.xlsx','TA');
  TB=@ole('D:\MCM2021\fact50.xlsx','TB');
  TC=@ole('D:\MCM2021\fact50.xlsx','TC');!the type sell by each factory;
  maxp=@ole('D:\MCM2021\fact50.xlsx','MAXP');!the max production of
each factory;
  limit=23406 42743 17053;!limit of A,B,C;
  ctrl=1;!0 is z1,1 is z2;
  g=? ?;!the z;
  @ole('D:\MCM2021\pro1buyfinal.xlsx','buy')=ans;!the answer;
enddata

!object;
z(1)=@sum(buy:f);!the min of factory amount;
z(2)=1.2*@sum(amount:xA)+1.1*@sum(amount:xB)+@sum(amount:xC)+2*@sum(a
```

```
mount:SA+SB+SC);!the min money to buy and store;
u=@sum(amount:SA+SB+SC);
min=(1-ctrl)*z(1)+ctrl*z(2);
!@sum(soft_variable:dm);
!(1-ctrl)*z(1)+ctrl*z(2);

!conditions;
SA(1)=0;
SB(1)=0;
SC(1)=0;!innitial value;
z(1)=36;
@sum(soft_variable:dm)<5338.147;!limit lack;
@for(amount(i)|i #ne#
1:SA(i)/0.6+SB(i)/0.66+SC(i)/0.72+dm(i)-dp(i)>28200*2);!we should
store for 2 weeks' production;
!for the first day;
xA(1)>uA(1);
xB(1)>uB(1);
xC(1)>uC(1);
@for(amount(i)|i #ne#
1:SA(i)=SA(i-1)+xA(i)-uA(i);SB(i)=SB(i-1)+xB(i)-uB(i);SC(i)=SC(i-1)+x
C(i)-uC(i));!the recursive relationship of S;
@for(amount(j):uA(j)/0.6 + uB(j)/0.66 +
uC(j)/0.72+dm(j)-dp(j)>28200);!the capability should be produced;
@for(amount(j):xA(j)+xB(j)+xC(j)<6000*8);!the goods can be translated;


@for(buy:@bin(f));
!conditions;
@for(amount(j):xA(j)=@sum(buy(i):mb(i,j)*f(i)*TA(i));xB(j)=@sum(buy(i)
:mb(i,j)*f(i)*TB(i));xC(j)=@sum(buy(i):mb(i,j)*f(i)*TC(i)));
!each week the amount from each factory of each type is enough;
@for(res(i,j):mb(i,j)<rdm(j)*maxp(i)+(1-rdm(j))*(mid(i)+maxp(i))/2);!
each factory has its limit;
@for(res(i,j):ans(i,j)=mb(i,j)*f(i));
!soft;
!u+dm(1)-dp(1)=779384.8;
@for(goal(i):z(i)<g(i));
End
```

多阶段优化

```
model:
sets:
  fact/1,2..36/:fbuy;!the amount bought from each company;
```

```
  trans/1,2..8/:pbrk;!the translation company;
  week/1,2..24/:w;!each week;
  ans(fact,trans):plan,pwhe,fplan;!how much,whether;
  supply(fact,week):s;!the amount in each week bought from each factory;
  lost(trans,week):lp;!the lost portion of each translator in each week;
  soft_var/1..9/:dm,dp;!soft variables;
endsets

data:
  s=@ole('D:\MCM2021\pro1buyfnl36.xlsx','buy');!the buying plan;
  lp=@ole('D:\MCM2021\comp.xlsx','PL');!the loss;
  ctrl=?;
  !@ole('D:\MCM2021\pro1translation.xlsx','week14')=fplan;
enddata
!object;
min=@sum(soft_var:dp);
!conditions;
!hard conditions;
@for(fact(i):fbuy(i)=s(i,ctrl));!the amount to buy in this week;
@for(trans(j):pbrk(j)=lp(j,ctrl));!the loss portions in this week;
@for(ans:@bin(pwhe));!whether 0-1;
@for(fact(i):@sum(trans(j):plan(i,j)*pwhe(i,j))=fbuy(i));!all buy to
be transfored;
@for(trans(j):@sum(fact(i):plan(i,j)*pwhe(i,j))<6000);!one translator
cannot afford more than 6000;
!soft_condition;
@for(trans(j):@sum(fact(i):pwhe(i,j))+dm(j)-dp(j)=1);!one suppler best
have one translator;
@sum(fact(i):@sum(trans(j):plan(i,j)*pbrk(j)))+dm(9)-dp(9)=0;!the min
lost;
@for(ans:fplan=plan*pwhe);
End
```

问题三购买和运输方案

```
model:
sets:
  !produce and buy;
  week/1..24/:xA,xB,xC,SA,SB,SC,uA,uB,uC;!the amount to buy each
week,the store in each week,the use in each week;
  factory/1..50/:maxp,mid,TA,TB,TC;!factories max,midium production and
its type;
  buyplan(factory,week):x,rdm;!the buy plan in each factory in each
week,the random change in each week;
  !translation and loss;
```

```
  translator/1..8/:tr;!the translators;
  ploss(translator,week):lp;!the loss portion of each translator in each
week;
  tranplan(factory,translator,week):y,pwhe;!how much to translate and
whether choose;
  tplanw(factory,translator):tpw;!fixed week plan;
  soft_con/1..123/:dm,dp;!the soft conditions;
  goal/1..5/:z;!the goalsin each preference;
endsets

data:
  rdm=@qrand(123);
  mid=@ole('D:\MCM2021\mid.xlsx','mid');!the midiam;
  TA=@ole('D:\MCM2021\factoryinfo.xlsx','TA');
  TB=@ole('D:\MCM2021\factoryinfo.xlsx','TB');
  TC=@ole('D:\MCM2021\factoryinfo.xlsx','TC');!the type sell by each
factory;
  maxp=@ole('D:\MCM2021\factoryinfo.xlsx','MAXP');!the max production
of each factory;
  lp=@ole('D:\MCM2021\comp.xlsx','PL');!the loss;
enddata

!object;
min=10*z(1)+5*(z(2)+z(3))+3*z(4);
z(1)=dp(97)+dp(98)+dp(99);!min price;
z(2)=@sum(week(j):dm(j)+dp(j+24));!more A less C;
z(3)=@sum(week(j):dp(j+48));!less lost;
z(4)=@sum(week(j):dm(j+99));!try ro store for 2 weeks;
!z(5)=@sum(week(j):dp(j+72));!one fact should try to have only one
translator;

!conditions;
!hard conditions;
!production;
SA(1)=0;SB(1)=0;SC(1)=0;!innitial value;
@for(week(i)|i #ne# 1:SA(i)/0.6+SB(i)/0.66+SC(i)/0.72>28200*2);!we
should store for 2 weeks' production;
!for the first day;
xA(1)=uA(1);xB(1)=uB(1);xC(1)=uC(1);
@for(week(i)|i #ne#
1:SA(i)=SA(i-1)+xA(i)-uA(i);SB(i)=SB(i-1)+xB(i)-uB(i);SC(i)=SC(i-1)+x
C(i)-uC(i));!the recursive relationship of S;
@for(week(j):xA(j)=@sum(factory(i):x(i,j)*TA(i));xB(j)=@sum(factory(i)
:x(i,j)*TB(i));xC(j)=@sum(factory(i):x(i,j)*TC(i)));
```

```
@for(week(j):@for(factory(i):x(i,j)<(1-rdm(i,j))*(maxp(i)+mid(i))/2+r
dm(i,j)*maxp(i)));!one factory has its limit;

!translation;
!@for(tranplan:@bin(pwhe));!whether choose this translator in this week;
@for(week(k):@for(translator(j):@sum(factory(i):y(i,j,k))<6000));!one
translator cannot afford 60000;
@for(week(k):@for(factory(i):@sum(translator(j):y(i,j,k))=x(i,k)));!t
he goods should be translated;

!soft conditions;
!prices;
@for(week(j):uA(j)/0.6 + uB(j)/0.66 +
uC(j)/0.72+dm(j+99)-dp(j+99)>28200);!the capability should be prepared;
1.2*@sum(week:xA)+1.1*@sum(week:xB)+@sum(week:xC)+dm(97)-dm(97)=52803
7.3;!bought fee;
@sum(week:SA+SB+SC)+dm(98)-dp(98)=779384.8;!store fee;
@sum(week:xA+xB+xC)+dm(99)-dp(99)=442676.8;!translation fee;

!more A,less C;
@for(week(j):xA(j)+dm(j)-dp(j)=28200*0.6);
@for(week(j):xC(j)+dm(j+24)-dp(j+24)=0);

!min loss;
@for(week(k):@sum(tplanw(i,j):y(i,j,k)*lp(j,k))+dm(k+48)-dp(k+48)=0);

!try to finish in one translator per fact;
@for(tranplan(i,j,k):pwhe(i,j,k)=@if(y(i,j,k)#gt#1,1,0));!calculate
pwhe;
@for(week(k):@for(factory(i):@sum(translator(l):pwhe(i,l,k))+dm(k+72)
-dp(k+72)=1));
End
```

问题四最大产能

```
model:
sets:
 !produce and buy;
  week/1..24/:xA,xB,xC,SA,SB,SC,uA,uB,uC,tmax;
!the amount to buy each week,the store in each week,the use in each week,the
max week translation;
  factory/1..119/:maxp,mid,TA,TB,TC;!factories max,midium production
and its type;
  stand/1..180/:maxpt,midt,TAt,TBt,TCt;
  buyplan(factory,week):x,rdm;!the buy plan in each factory in each
```

```
week,the random change in each week;
  !translation and loss;
  translator/1..8/:tr;!the translators;
  ploss(translator,week):lp;!the loss portion of each translator in each
week;
  !soft_conditions;
  soft_var/1..3/:dm,dp;
  tranplan(factory,translator,week):y;!how much to translate;

endsets

data:
  rdm=@qrand(123);
  midt=@ole('D:\MCM2021\TOPSISall1.xlsx','mid');!the midiam;
  TAt=@ole('D:\MCM2021\TOPSISall1.xlsx','TA');
  TBt=@ole('D:\MCM2021\TOPSISall1.xlsx','TB');
  TCt=@ole('D:\MCM2021\TOPSISall1.xlsx','TC');!the type sell by each
factory;
  maxpt=@ole('D:\MCM2021\TOPSISall1.xlsx','MAXP');!the max production
of each factory;
  lp=@ole('D:\MCM2021\comp.xlsx','PL');!the loss;
  tmax=@ole('D:\MCM2021\maxtran.xlsx','tmax');!the max week tran;
  @ole('D:\MCM2021\pro4buy1.xlsx','buy')=x;!get the answer in xlsx;
enddata

@for(factory(i):maxp(i)=maxpt(i);mid(i)=midt(i);TA(i)=TAt(i);TB(i)=TB
t(i);TC(i)=TCt(i));

!object;
max=c-(0.2*dm(1)+0.1*(dm(2)+dp(3)));!max production;

!conditions;
!hard conditions;
!production;
SA(1)=0;SB(1)=0;SC(1)=0;!innitial value;
@for(week(i)|i #ne#
1:SA(i)/0.6+SB(i)/0.66+SC(i)/0.72+dm(1)-dp(1)=2*c);!we should store
for 2 weeks' production;
!for the first day;
xA(1)>uA(1);xB(1)>uB(1);xC(1)>uC(1);
@for(week(i)|i #ne#
1:SA(i)=SA(i-1)+xA(i)-uA(i);SB(i)=SB(i-1)+xB(i)-uB(i);SC(i)=SC(i-1)+x
C(i)-uC(i));!the recursive relationship of S;
@for(week(j):xA(j)=@sum(factory(i):x(i,j)*TA(i));xB(j)=@sum(factory(i)
```

```
:x(i,j)*TB(i));xC(j)=@sum(factory(i):x(i,j)*TC(i)));
@for(week(j):@for(factory(i):x(i,j)<maxp(i)*rdm(i,j)+(mid(i)+maxp(i))
/2*rdm(i,j)));!one factory has its limit;
@for(week(j):uA(j)/0.6 + uB(j)/0.66 + uC(j)/0.72=c);!the capability
should be prepared;
!translation;
!@for(tranplan:@bin(pwhe));!whether choose this translator in this week;
@for(week(k):@for(translator(j):@sum(factory(i):y(i,j,k))<6000));!one
translator cannot afford 60000;
@for(week(k):@for(factory(i):@sum(translator(j):y(i,j,k))=x(i,k)));!t
he goods should be translated;

!more A,less C;

@for(week(j):xA(j)+dm(2)-dp(2)=0.6*c);
@for(week(j):xC(j)+dm(3)-dp(3)=0);
@for(week(j):xA(j)+xB(j)+xC(j)<tmax(j));!max week tran;
```

遗传算法

```
global fbuy;%最省钱购买计划
global lossp;%损失率
fbuy=xlsread('D:\MCM2021\supply3.xlsx','A1:X50');
lossp=xlsread('D:\MCM2021\comp.xlsx','B2:Y9');
%function [x,fval]=genes(fun,lb,ub,epslon,conditions,index)
%遗传算法函数，输入适应度函数，下界，上界，精度，条件函数默认<=0，参数数组（最
大进化代数，种群大小,交叉概率，变异概率）
%记住！最大进化数要够大！种群大小也是！主要是交叉概率在起作用！交叉概率要0.5
以上

X=xlsread('D:\MCM2021\supply3.xlsx','A1:X50');
%%
week=3;%第一周
%%
 index=[200,100,0.7,0.01];
  G=index(1);%最大进化代数
  NP=index(2);%种群大小
  pexc=index(3);%交叉概率
  pdiv=index(4);%变异概率

  %上界是第k期对每一个供应商的购买量
  ub=X(:,week);
  ub=ub';
  %下界是0
  lb=zeros(1,size(X,1));
```

```
%%
%维度
L=50;
%我们采取8*50的形式表示一次运输
fvalbest=zeros(1,G);%初始化最适应值
Xbest=zeros(G,8*L);%初始化最优个体

k=1;%第一代

%产生初始种群
group=rand(NP*8,L).*(ub-lb);%产生NP个个体的基因,连着的8个代表一个个体


%计算NP个个体的适应度
fit=zeros(NP,1);

for k=1:G
    %计算适应度
     index=1:8;
    for i=1:NP
        temp=group(index+(i-1)*8,:);
        %调整group,使得运输公司加和更容易小于6000且单个供给方的运输公司尽
量少
        temp=adjust(temp,ub);
        group(index+(i-1)*8,:)=temp;
        more=sum(temp,2)-6000;
        punish=7*length(find(more>0));%不超过最大运输量的限制
        %if conditions ~= [] %如果有罚函数
            %punish=conditions(indiv);%罚函数
        %end
        fit(i)=fitfun(temp,week)-punish;%实数数组计算适应度
    end

    %找到初始最适应个体
    fvalbest(k)=max(fit);
    maxplace=find(fit==max(fit));
    bestmat=group(index+maxplace(1),:);
    Xbest(k,:)=bestmat(:)';


%%
```

```
    %赌轮盘复制个体
    stand=cumsum(fit)./sum(fit);%计算概率标准
    for i=1:NP
        p=rand;%得到一次概率
        flag=find(stand>p);%找到第一个大于p的
        if double(isempty(flag))==1
            ch=NP;
        else
            ch=flag(1);
        end
        newG((i-1)*8+index,:)=group((ch-1)*8+index,:);%复制
    end
%%

    %交叉
    chosen=double(rand(1,NP)<pexc);%得到是否选择该个体的数组
    chosenplace=find(chosen==1);%被选择的位置
    if mod(length(chosenplace),2)==1%不是成对的
        %从没被选中的挑一个
        nchosenplace=find(chosen==0);
        newch=round(1+(length(nchosenplace)-1)*rand);
        chosen(nchosenplace(newch))=1;
        chosenplace=find(chosen==1);%更新一下
    end
    %%
    %现在保证成对了
    for i=1:2:length(chosenplace)
        exc1=newG((chosenplace(i)-1)*8+index,:);
        exc2=newG((chosenplace(i+1)-1)*8+index,:);%找到一对
        excplace=double(rand(1,L)<pexc);%哪个供应商的运输方案交换，保证落
入的范围还是符合标准
        tempexc=exc1(:,find(excplace==1));%拷贝交换位置

        tranplace=double(rand(1,8)<pexc);%运输商分配交换
        tranexp=find(tranplace==1);
        temptran=exc1(tranexp,:);
        exc1(tranexp,:)=exc2(tranexp,:);
        exc2(tranexp,:)=temptran;

        exc1(:,find(excplace==1))=exc2(:,find(excplace==1));
        exc2(:,find(excplace==1))=tempexc;%完成交换


        %更改种族
```

```
            newG((chosenplace(i)-1)*8+index,:)=exc1;
            newG((chosenplace(i+1)-1)*8+index,:)=exc2;
        end
        %%
       %变异
        chdiv=double(rand(1,NP)<pdiv);
        divplace=find(chdiv==1);%找到变异位置
        for j=1:length(divplace)
           %每一个变异个体，哪些位置变异
            temp=newG((divplace(j)-1)*8+index,:);
            dex=find(sum(temp)>0);%被购买的公司序号
            divpoint=find(double(rand(1,length(dex))<pdiv)==1);%找到变异非0位
置
            %实际变异位置
            diverpt=dex(divpoint);
            if double(isempty(diverpt))==0

newG((divplace(j)-1)*8+index,diverpt)=rand(8,length(diverpt)).*(ub(diverpt)
-lb(diverpt));%变异
            end
        end
        %%
        group=newG;%更新种群

 end
```

适应函数

```
function fit=fitfun(X,week)%目标函数
  global fbuy;%最省钱购买计划
  global lossp;%损失率
  bplan=fbuy(:,week);
  lp=lossp(:,week);

  %损失
  F1=sum(sum(lp.*X))*0.001;

  %最好由一家运
  temp=double(X~=0);%哪些有运
  comp=sum(temp,2);%一个列向量，表示每家由几家运
  F2=length(find(comp>1));%超过一家的量

  fit=100/(F1+F2);%适应度

end
function newtemp=adjust(temp,ub)
```

```
    %%
    dex=find(sum(temp)>0);%被购买的公司序号
    [~,rank]=sort(temp(:,dex));
    %由多少个公司运转
    m=floor(3*rand)+4;
    for i=1:length(dex)
        temp(rank(1:m,i),dex(i))=0;%把那种小小的值去掉

temp(rank(6:8,i),dex(i))=temp(rank(6:8,i),dex(i))./sum(temp(rank(6:8,i),dex
(i)));
        temp(rank(6:8,i),dex(i))=temp(rank(6:8,i),dex(i)).*ub(dex(i));%使
得加起来是购买量
    end
    newtemp=temp;
end
```

指标计算

```
indice_ls=[]
for i in range(402):
    pur_i = pur[i]
    sup_i = sup[i]
    pur_count = np.nonzero(pur_i)[0].shape[0]  #下单次数
    sup_count = np.nonzero(sup_i)[0].shape[0]  #供货次数
    feedback_ratio = sup_count / pur_count  #订单响应比率
    avg_feedback_time, avg_weighted_feedback_time, default_nums,
feedback_time_ls = cal_feedback_time(
        pur_i, sup_i)  #平均响应时间，平均加权响应时间,未响应的订单数，订
单响应时间序列

    '''#计算平均到货率，10 个生产周期的到货率序列，
    &以及每个有实际供货的生产周期的*最大周供应量的平均值*和*最大周供
应量序列*'''
    avg_arrival_ratio, arrival_ratio_ls, avg_max_period_weeksup,
max_period_weeksup_ls, SYI, CV = cal_period(
        pur[i], sup[i])#6

    max_week_sup = sup[i].max()  #五年最大周供应量
    median_week_sup = np.median(sup[i][np.nonzero(sup[i])])  #五年供应量的中
位数

    indicators=(pur_count, sup_count, #2
                feedback_ratio, avg_feedback_time,
avg_weighted_feedback_time, default_nums, avg_arrival_ratio, #5
                max_week_sup,  median_week_sup,avg_max_period_weeksup,
#3
                SYI, CV,#2
               feedback_time_ls, arrival_ratio_ls, max_period_weeksup_ls)#3

    indice_ls.append(indicators )


def cal_period(pur_i,sup_i):
    pur_period=[]
    sup_period=[]
    max_period_weeksup_ls=[]
    for i in range(10):
        pur_year_i = pur_i[24*i:24*(i+1)]
        sup_year_i = sup_i[ 24*i:24*(i+1)]

        pur_period.append(pur_year_i.sum())
        sup_period.append(sup_year_i.sum())
```

```
            max_period_weeksup_ls.append(sup_year_i.max())

        pur_period=np.array(pur_period)
        sup_period=np.array(sup_period)
        max_period_weeksup_ls=np.array(max_period_weeksup_ls)



avg_max_period_weeksup=np.array(max_period_weeksup_ls)[np.nonzero(sup_period)].mean()#去掉无供货的生产周期
        arrival_ratio_ls=sup_period/pur_period
        avg_arrival_ratio=arrival_ratio_ls[np.nonzero(pur_period)].mean()#计算平均到货率时，去掉无订单的生产周期


SYI=(sup_period[np.nonzero(pur_period)].mean()-sup_period[np.nonzero(pur_period)].std())/sup_period[np.nonzero(pur_period)].max()

CV=sup_period[np.nonzero(pur_period)].std()/sup_period[np.nonzero(pur_period)].mean()

        return avg_arrival_ratio, arrival_ratio_ls, avg_max_period_weeksup, max_period_weeksup_ls,SYI,CV



def cal_feedback_time(pur_i,sup_i):
        pur_non=np.nonzero(pur_i)[0]#np.nonzero()返回的是 tuple 套 array，需要手动变为 array
        sup_non=np.nonzero(sup_i)[0]#np.nonzero()返回的是 tuple 套 arra，需要手动变为 array
        feedback_time_ls=[]
#         print(pur_non)
        for i in range(len(pur_non)):
            pur_non_i=pur_non[i]

            '''求订单 i 对应的最近的供给 i 的时间
            存在多个订单对应一个供给的情况'''
#             print(pur_non_i)
            a=sup_non -pur_non_i
            b=np.where(a>=0,True,False)
            a=a[b]
            if max(sup_non)<pur_non_i:#如果订单日期后无供货响应
                feedback_time_i=-1#如果订单迄今没有响应，则响应时间记为-1
```

```
                #np.inf#如果订单迄今没有响应，则响应时间记为正无穷
            else:
                feedback_time_i=min(a)

            feedback_time_ls.append(feedback_time_i)

        feedback_time_ls=np.array(feedback_time_ls)
        weighted_feedback_time_ls=feedback_time_ls*pur_i[pur_non]

        default_nums=np.where(feedback_time_ls<0,1,0).sum()#记录没有响应的订单
次数

avg_feedback_time=feedback_time_ls[np.where(feedback_time_ls>=0,True,False)].m
ean()

avg_weighted_feedback_time=weighted_feedback_time_ls[np.where(feedback_time
_ls>=0,True,False)].mean()
#         print(feedback_time_ls)
        return
avg_feedback_time,avg_weighted_feedback_time,default_nums ,feedback_time_ls



        def cal_fb_sup_time(s=sup[0],i=42):
            a=np.array(np.nonzero(s)) -i
            b=np.where(a>=0,True,False)
            if b==np.array([]):
                        sup_non_i=np.inf#如果订单迄今没有响应，则响应时间记为正
无穷
            a=a[b]
            sup_non_i=min(a)
            return sup_non_i

        def tral():
            pur_non=np.nonzero(pur[0])[0]#np.nonzero()返回的是 tuple 套 array，需
要手动变为 array
            sup_non=np.nonzero(sup[0])[0]#np.nonzero()返回的是 tuple 套 arra，需
要手动变为 array
            for i in range(len(pur_non)):
                pur_non_i=pur_non[i]
                print(cal_fb_sup_time(sup[0],pur_non_i))
        tral()


df_indice = pd.DataFrame(data=np.c_[names,types,indice_array],
```

```
                                columns=['供货商 ID','材料分类',
                                    '下单周次数',
                                    '供货次数',
                                    '订单响应比率',
                                    '平均响应时间',
                                    '加权平均响应时间',
                                    '沉没订单数',
                                    '平均到货率',
                                    '五年最大周供应量', '五年周供应量的中位数','
五年生产周期内最大周供应量的平均值', 'SYI', 'CV',
                                    '五年响应时间序列', '五年到货率序列', '五年生
产周期内的最大周供应量序列'
                                ])
# df_indice.to_excel('附件 1 指标.xlsx',index=False)
df_indice
```

指标筛选

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
#  当 VIF<10,说明不存在多重共线性；当 10<=VIF<100,存在较强的多重共线性，当
VIF>=100,存在严重多重共线性
x=df_only_ind
x1=df_only_ind.iloc[:,2:-1]
x2=df_only_ind.iloc[:,3:-1]
vif = [variance_inflation_factor(x.values, x.columns.get_loc(i)) for i in x.columns]
Vif

from statsmodels.stats.outliers_influence import variance_inflation_factor
tol = [1./variance_inflation_factor(x.values, x.columns.get_loc(i)) for i in x.columns]
Tol


plt.figure(figsize=(12,8))
sns.heatmap(df_only_ind.corr(),annot=True)
```

Decompose 数据分解

```
# def get_decom_data():
decom_data_ls=[]
for i in [0,1,5,6,7]:
    decomposition=seasonal_decompose(tran[i],period=24)
    trend = decomposition.trend #趋势效应
```

```
        seasonal = decomposition.seasonal #季节效应
        residual = decomposition.resid #随机效应
        data_i_array=np.c_[trend,seasonal,residual].T
        decom_data_ls.append(data_i_array)
# for i in range(len(decom_data_ls)):
decom_data_array=np.r_[decom_data_ls[0],decom_data_ls[1],decom_data_ls[2],de
com_data_ls[3],decom_data_ls[4] ]
np.isnan(decom_data_array).sum(axis=1)

from statsmodels.tsa.seasonal import seasonal_decompose
# plt.figure(figsize=(12,20))
result1 = seasonal_decompose(tran[0],period=24)
# result1.trend.shape,result1.seasonal.shape,
np.c_[result1.trend,result1.seasonal].shape

def draw_decom(tran_i):
        decomposition=seasonal_decompose(tran_i,period=24)
        trend = decomposition.trend #趋势效应
        seasonal = decomposition.seasonal #季节效应
        residual = decomposition.resid #随机效应
        plt.figure(figsize=(16,8))
        plt.subplot(411)
        plt.plot(tran_i, label=u'原始数据')
        plt.legend(loc='best')
        plt.xlabel('周')
#         plt.ylabel('损耗率%')
        plt.subplot(412)
        plt.plot(trend, label=u'趋势')
        plt.legend(loc='best')
        plt.xlabel('周')
#         plt.ylabel('损耗率%')

        plt.subplot(413)
        plt.plot(seasonal,label=u'季节性')
        plt.legend(loc='best')
        plt.xlabel('周')
#         plt.ylabel('损耗率%')

        plt.subplot(414)
        plt.plot(residual, label=u'残差')
        plt.legend(loc='best')
        plt.xlabel('周')
#         plt.ylabel('损耗率%')
```

```
    plt.tight_layout()
    #return np.c_[trend, seasonal, residual].T

draw_decom(tran[0])


def cal_tran3_period(tran_3):
    tran_3=tran_3.copy()
    tran_3[tran_3==max(tran_3)]=0
    period=[]
    tran_3_i=tran_3[:24]
    for i in range(1,10):
        tran_3_i = np.c_[tran_3_i,tran_3[24*i:24*(i+1)]]
#       plt.plot(tran_3_i)
    tran_3_i=tran_3_i.T
    avg_week=tran_3_i.sum(axis=0)/np.count_nonzero(tran_3_i,axis=0)
    plt.plot(avg_week)
    return tran_3_i ,avg_week
cal_tran3_period(tran_3)[1]


def draw2(tran_3):
    plt.figure(figsize=(16,6))
    for i in range(10):
            tran_3_i = tran_3[24*i:24*(i+1)]
    #           tran_3_i=tran_3_i[np.nonzero(tran_3_i)]
            plt.subplot(121)
            plt.scatter(range(1,25),tran_3_i,label='生产周期{}'.format(i+1))
#               plt.legend()
            plt.subplot(122)
            plt.plot(range(1,25),tran_3_i,label='生产周期{}'.format(i+1))
#               plt.legend()
    plt.show()

draw2(tran_3)
```

数据探索与可视化

```
A=df_pur[df_pur['材料分类']=='A'].iloc[:,2:].values
A_year=A[:, :24]
```

```
# Period=[]
# Period.append(A_year.sum())

# for i in range(1,10):
#         A_year +=A[:, 24*i:24*(i+1)]
#         Period.append(A[:,24*i:24*(i+1)].sum())
# A_year,Period

def cal_year(A):
        A_year=A[:, :24]
        Period=[]
        Period.append(A_year.sum())

        for i in range(1,10):
                A_year +=A[:, 24*i:24*(i+1)]
                Period.append(A[:,24*i:24*(i+1)].sum())
        return A_year,np.array(Period)
A_year,A_period =cal_year(A)


def draw(A_year):
        A_year_std=A_year.std(axis=0)
        A_year_mean=A_year.mean(axis=0)

        x=range(1,25)
        plt.figure(figsize=(15,6))


        plt.subplot(131)
        plt.plot(x,A_year_std,label='std')
        plt.title('std')
        plt.xlim((1,24))

        plt.subplot(132)
        plt.plot(x,A_year_mean,label='mean')
        plt.title('mean')
        plt.xlim((1,24))

        plt.subplot(133)
        plt.plot(x,A_year.sum(axis=0),label='mean')
        plt.title('sum')
        plt.xlim((1,24))

        plt.show()
```

```
draw(A_year)


B=df_pur[df_pur['材料分类']=='B'].iloc[:,2:].values

def cal_year(A):
    A_year=A[:, :24]
    Period=[]
    Period.append(A_year.sum())

    for i in range(1,10):
        A_year +=A[:, 24*i:24*(i+1)]
        Period.append(A[:,24*i:24*(i+1)].sum())
    return A_year,np.array(Period)
B_year,B_period =cal_year(B)
draw(B_year)

C=df_pur[df_pur['材料分类']=='C'].iloc[:,2:].values

def cal_year(A):
    A_year=A[:, :24]
    Period=[]
    Period.append(A_year.sum())

    for i in range(1,10):
        A_year +=A[:, 24*i:24*(i+1)]
        Period.append(A[:,24*i:24*(i+1)].sum())
    return A_year,np.array(Period)
C_year,C_period =cal_year(C)
draw(C_year)

plt.plot(range(1,11),Period)
plt.title('十个生产周期总订单的的逐期变化')
plt.show()



x=range(1,11)
for i in range(3):
    plt.plot(x,[j[i] for j in S_period],label="供应商 S00{}".format(i+1))
plt.legend()
plt.show()
```

```
plt.figure(figsize=(16,6))
plt.plot(range(1,241),pur.sum(axis=0),label='订货量')
plt.plot(range(1,241),sup.sum(axis=0),label='供货量')
plt.legend()
plt.xlabel('周')
plt.ylabel('原材料数量$m^3$')



plt.figure(figsize=(16,6))
plt.plot(range(1,25),pur_year.mean(axis=0),label='订货量')
plt.plot(range(1,25),sup_year.mean(axis=0),label='供货量')
plt.legend()
plt.xticks(range(25),range(25))
plt.grid(axis='x')
plt.xlabel('周')
plt.ylabel('原材料数量$m^3$')


stock_A=A_sup.sum(axis=0)
stock_B=B_sup.sum(axis=0)
stock_C=C_sup.sum(axis=0)
stock_ABC=stock_A+stock_B+stock_C


#订货量
plt.figure(figsize=(16,6))
plt.plot(cal_year(A)[0].mean(axis=0),label='A')
plt.plot(cal_year(B)[0].mean(axis=0),label='B')
plt.plot(cal_year(C)[0].mean(axis=0),label='C')
plt.legend()


#供货量
plt.figure(figsize=(16,6))
plt.plot(cal_year(A_sup)[0].mean(axis=0),label='A')
plt.plot(cal_year(B_sup)[0].mean(axis=0),label='B')
plt.plot(cal_year(C_sup)[0].mean(axis=0),label='C')
plt.legend()


plt.figure(figsize=(16,6))
plt.plot(stock_A/stock_ABC,label='A')
plt.plot(stock_B/stock_ABC,label='B')
```

```
plt.plot(stock_C/stock_ABC,label='C')
plt.legend()
# plt.axhline(y=2*28200,c='r')
# plt.axhline(y=stock_ABC.mean(),c='y')#平均库存量
# plt.ylabel('库存量')


def cal_stock(worn_rate=0.025, prod=prod):
    stock_in=28200*2#假设已有库存为两周产能，即 2*2822
#     stock_ls=[]
#     worn_rate=0.025
    stock_worn=28200*2#假设已有库存为两周产能，即 2*2822
    stock_worn_ls=[]


    gap_ls=[]
    for i in prod:
        stock_in+=i
        stock_worn+=i*(1-worn_rate)
        gap=0
        if stock_worn-28200<0:
            gap=stock_worn-28200
            stock_worn=0
#         elif stock_in-28200<0:
#             stock_in=0
        else:
#             stock_in=stock_in-28200
            stock_worn=stock_worn-28200
            gap=0
#         stock_ls.append(stock_in)#库存量

        stock_worn_ls.append(stock_worn)#考虑转运损耗后的库存量
        gap_ls.append(gap)
#     stock_array=np.array(stock_ls)

    stock_worn_array=np.array(stock_worn_ls)
    gap_array=np.array(gap_ls)
    return stock_worn_array,gap_array
np.count_nonzero(cal_stock()[0])


def cal_stock_1(prod=prod):
#     stock_in=28200*2#假设已有库存为两周产能，即 2*2822
#     stock_ls=[]
```

```
#         worn_rate=0.025
        stock_worn=28200*2#假设已有库存为两周产能，即 2*2822
        stock_worn_ls=[]

        gap_ls=[]
        for i in range(240):
#             stock_in+=
            worn_rate=avg_week_worn[i]/100
            stock_worn+=prod[i]*(1-worn_rate)
            gap=0
            if stock_worn-28200<0:
                gap=stock_worn-28200
                stock_worn=0
#             elif stock_in-28200<0:
#                 stock_in=0
            else:
#                 stock_in=stock_in-28200
                stock_worn=stock_worn-28200
                gap=0
#             stock_ls.append(stock_in)#库存量

            stock_worn_ls.append(stock_worn)#考虑转运损耗后的库存量
            gap_ls.append(gap)
#         stock_array=np.array(stock_ls)

        stock_worn_array=np.array(stock_worn_ls)
        gap_array=np.array(gap_ls)
        return stock_worn_array,gap_array
np.count_nonzero(cal_stock()[0])

plt.figure(figsize=(16,6))
plt.plot(range(1,241),(cal_stock_1()[1]+28200)/28200,c='#F5B14C')
plt.ylabel('产能利用率')
plt.xticks([1,50,100,150,200,240],[1,50,100,150,200,240])
plt.xlim((1,240))
plt.ylim((0.6,1.01))
plt.fill_between(range(1,241),          0.6,          (cal_stock_1()[1]+28200)/28200,
facecolor='#F5B14C', alpha=0.3)

plt.axhline(y=0.9804187153020208,c='#661D98')
plt.annotate(
'总产能利用率=98.04%',#  注释的文字
xy=(110,0.97),#  箭头箭尖的位置
 xytext=(122,0.92),#  注释文字的位置
```

```
color = "black",#  文字的颜色
fontsize = 15,#  文字的尺寸
arrowprops=dict(facecolor='black', shrink=0.001))

plt.show()

plt.figure(figsize=(16,6))
# plt.plot((cal_stock(0)[1]+28200)/28200,label='损耗率=0')
plt.plot((cal_stock(0.01)[1]+28200)/28200, label='损耗率=1%')
plt.plot((cal_stock(0.02)[1]+28200)/28200, label='损耗率=2%')
plt.plot((cal_stock(0.03)[1]+28200)/28200, label='损耗率=3%')
plt.legend(loc='best')
# plt.axhline(y=2*28200,c='r')
# plt.axhline(y=stock_array.mean(),c='y')#平均库存量
plt.ylabel('产能利用率')


plt.figure(figsize=(12,6))
plt.plot(cal_stock(0)[0],label='损耗率=0')
plt.plot(cal_stock_1()[0], label='损耗率=每周平均损耗率')
# plt.plot(cal_stock(0.02), label='损耗率=2%')
# plt.plot(cal_stock(0.03), label='损耗率=3%')
plt.legend()
plt.axhline(y=2*28200,c='r')
plt.annotate(text='两周生产量',xy=(160,28200*2),xytext=(165,40000),color='g',arrowprops=dict(facecolor='black',
shrink=0.5))
# plt.axhline(y=stock_array.mean(),c='y')#平均库存量
plt.ylabel('库存可生产的产品量')

plt.figure(figsize=(16,6))
plt.subplot(211)
plt.plot(np.cumsum(prod))#累计到货率——不计消耗
plt.plot(np.cumsum(28200*np.ones(240)),c='r')
plt.subplot(212)
plt.plot(np.cumsum(prod)-np.cumsum(28200*np.ones(240)),)
```

数据录入

```
def clean_trans(plan_trans_3):
#         week=1
        all_ls=[]
        for sup_name in range(1,plan_trans_3.iloc[:,0].nunique()+1):
```

```
#         plan_trans_3[plan_trans_3['周数']==week]
#         sup_name=1
            i_ls=[sup_name]
            for week in range(1,25):
                i_week_trans=plan_trans_3[plan_trans_3['周数']==week][plan_trans_3['供应商']==sup_name].iloc[:,-1].values
                i_ls.extend(i_week_trans)
                print(len(i_week_trans))
            all_ls.append(i_ls)
      all_array=np.array(all_ls)
      return all_array
clean_trans(plan_trans_3).shape

ranked_names=pd.read_excel('./剔除指标后/新供应商排名.xlsx').iloc[:,0]#.values
ranked_names.nunique()

rank2names=[(i,ranked_names[i]) for i in range(402)]
rank2names=dict(rank2names)
rank2names

pd.DataFrame(np.c_[np.arange(1,403),np.zeros((402,192))]).to_excel('转运方案 0 值底稿.xlsx',index=False)
cleand_tran3[0]=cleand_tran3[0].map(rank2names)
cleand_tran3.to_excel('cleaned 问题 3 转运方案(有供应商名称).xlsx',index=False,)

cleand_tran3.index=cleand_tran3[0]
cleand_tran3

tmpl=pd.read_excel('./转运方案 None 值底稿.xlsx')
tmpl.iloc[cleand_tran3.index-1,:]=cleand_tran3.values
tmpl.to_excel('问题 3 转运方案（正式版）.xlsx',index=False)
tmpl.iloc[cleand_tran3.index-1,:]

plan_trans_4=pd.read_excel('./plan/转运方案.xlsx',sheet_name='问题 4')
plan_trans_4

cleand_tran4=pd.DataFrame(clean_trans(plan_trans_4))#.to_excel('cleaned 问题 3 转运方案.xlsx',index=False)
cleand_tran4

cleand_tran4[0]=cleand_tran4[0].map(rank2names)
cleand_tran4#.to_excel('cleaned 问题 4 转运方案(有供应商名称).xlsx',index=False,)

cleand_tran4.index=cleand_tran4[0]
```

```
cleand_tran4


tmpl=pd.read_excel('./转运方案 None 值底稿.xlsx')
tmpl.iloc[cleand_tran4.index-1,:]=cleand_tran4.values
tmpl.to_excel('问题 4 转运方案（正式版）.xlsx',index=False)
tmpl.iloc[cleand_tran4.index-1,:]


plan_sup_2=pd.read_excel('./plan/副本 pro1buyfinal.xlsx').reset_index()
plan_sup_2.iloc[:,0]=plan_sup_2.iloc[:,0].map(rank2names)
plan_sup_2.index=plan_sup_2.iloc[:,0]
plan_sup_2


temp_sup=pd.read_excel('./供应方案 None 值底稿 .xlsx').iloc[:,:25]
# print(temp_sup.head(20))
temp_sup.iloc[plan_sup_2.index-1,:]=plan_sup_2#.values
temp_sup.to_excel('问题 2 供应方案（正式版）.xlsx',index=False)


plan_sup_3=pd.read_excel('./plan/pro3buyf.xlsx').reset_index()
plan_sup_3.iloc[:,0]=plan_sup_3.iloc[:,0].map(rank2names)
plan_sup_3.index=plan_sup_3.iloc[:,0]
plan_sup_3


temp_sup=pd.read_excel('./供应方案 None 值底稿 .xlsx').iloc[:,:25]
# print(temp_sup.head(20))
temp_sup.iloc[plan_sup_3.index-1,:]=plan_sup_3#.values
temp_sup.to_excel('问题 3 供应方案（正式版）.xlsx',index=False)


plan_sup_4=pd.read_excel('./plan/pro4buyfinal.xlsx').reset_index()
plan_sup_4.iloc[:,0]=plan_sup_4.iloc[:,0].map(rank2names)
plan_sup_4.index=plan_sup_4.iloc[:,0]
plan_sup_4


temp_sup=pd.read_excel('./供应方案 None 值底稿 .xlsx').iloc[:,:25]
# print(temp_sup.head(20))
temp_sup.iloc[plan_sup_4.index-1,:]=plan_sup_4#.values
temp_sup.to_excel('问题 4 供应方案（正式版）.xlsx',index=False)
```

实施方案评价

```
name2type=dict(zip(np.arange(1,403),types))
name2type

def cal_stock_1(prod=prod,avg_week_worn=avg_week_worn,init=28200*2):
#         stock_in=28200*2#假设已有库存为两周产能，即 2*2822
#         stock_ls=[]
#         worn_rate=0.025
        stock_worn=init#假设已有库存为两周产能，即 2*2822
        stock_worn_ls=[]

        gap_ls=[]
        for i in range(prod.shape[0]):
#             stock_in+=
            worn_rate=avg_week_worn[i]/100
            stock_worn+=prod[i]*(1-worn_rate)
            gap=0
            if stock_worn-28200<0:
                gap=stock_worn-28200
                stock_worn=0
#             elif stock_in-28200<0:
#                 stock_in=0
            else:
#                 stock_in=stock_in-28200
                stock_worn=stock_worn-28200
                gap=0
#             stock_ls.append(stock_in)#库存量

            stock_worn_ls.append(stock_worn)#考虑转运损耗后的库存量
            gap_ls.append(gap)
#         stock_array=np.array(stock_ls)

        stock_worn_array=np.array(stock_worn_ls)
        gap_array=np.array(gap_ls)
        return stock_worn_array,gap_array
np.count_nonzero(cal_stock()[0])

worn=pd.read_excel('损耗率.xlsx').iloc[:,1:]
worn.shape

remark_p_2=pd.read_excel('./plan/问题 2 供应方案（正式版）.xlsx')
remark_p_2.iloc[:,0]=remark_p_2.iloc[:,0].map(name2type)
```

```
p2A=remark_p_2[remark_p_2.iloc[:,0]=='A'].fillna(0).values[:,1:]
p2B=remark_p_2[remark_p_2.iloc[:,0]=='B'].fillna(0).values[:,1:]
p2C=remark_p_2[remark_p_2.iloc[:,0]=='C'].fillna(0).values[:,1:]
p2=remark_p_2.fillna(0).values[:,1:]
p2

plt.figure(figsize=(12,6))
# plt.plot(cal_stock(0)[0],label='损耗率=0')
plt.plot(range(1,25),y,label='实时库存量')

#cal_stock_1(prod2,worn_2)[0]) #label='损耗率=每周平均损耗率')
# plt.plot(cal_stock(0.02), label='损耗率=2%')
# plt.plot(cal_stock(0.03), label='损耗率=3%')
# plt.legend()
plt.axhline(y=2*28200,c='r')
plt.annotate(
'满足两周生产的库存量',# 注释的文字
xy=(11,2*28200),# 箭头箭尖的位置
  xytext=(12,2*26000),# 注释文字的位置
color = "black",# 文字的颜色
fontsize = 12,# 文字的尺寸
arrowprops=dict(facecolor='black', shrink=0.001))
# plt.axhline(y=stock_array.mean(),c='y')#平均库存量

plt.xticks(range(25),np.arange(25))
plt.ylabel('库存可生产的产品量')
plt.grid(axis='x')
plt.legend()


#采购花费， 1.2, 1.1, 1
p2A.sum()*1.2+p2B.sum()*1.1+p2C.sum()*1.1


t21 = []
for i in range(24):
      t21.append(
            (
#                (t2.iloc[:, 1 + i*8:8 * i + 9].values)*(worn.iloc[:, i + 1].values/100).sum()
                  (t2.iloc[:,i*8:8*(i+1)].sum(axis=0).values*worn.iloc[:,i]).sum()/100

      )
      )
t21=np.array(t21)
```

```
print(t21.shape)


#总损耗量 5502.493
#总量为 457525.63622437755
#整体损耗率 0.012026635220626317
t21.sum(),t2.sum().sum(),t21.sum()/457525.63622437755

#每周损耗量
t21.sum(axis=1),t21.sum()#,t2.iloc[:,1:].sum()
#总损耗量为 550249.3930955562 立方米


#每周损耗率
worn_2=t21/p2.sum(axis=0)
worn_2

p3.iloc[:,0]=p3.iloc[:,0].map(name2type)
p3A=p3[p3.iloc[:,0]=='A'].fillna(0).values[:,1:]
p3B=p3[p3.iloc[:,0]=='B'].fillna(0).values[:,1:]
p3C=p3[p3.iloc[:,0]=='C'].fillna(0).values[:,1:]
p3=p3.fillna(0).values[:,1:]
p3

plt.figure(figsize=(12,6))
# plt.plot(cal_stock(0)[0],label='损耗率=0')
plt.plot(range(1,25),cal_stock_1(prod3,worn_2,init=0)[0],label='实时库存量')

#cal_stock_1(prod2,worn_2)[0]) #label='损耗率=每周平均损耗率')
# plt.plot(cal_stock(0.02), label='损耗率=2%')
# plt.plot(cal_stock(0.03), label='损耗率=3%')
# plt.legend()
plt.axhline(y=2*28200,c='r')
plt.annotate(
'满足两周生产的库存量',#  注释的文字
xy=(11,2*28200),#  箭头箭尖的位置
  xytext=(12,2*26000),#  注释文字的位置
color = "black",#  文字的颜色
fontsize = 12,#  文字的尺寸
arrowprops=dict(facecolor='black', shrink=0.001))
# plt.axhline(y=stock_array.mean(),c='y')#平均库存量

plt.xticks(range(25),np.arange(25))
plt.ylabel('库存可生产的产品量')
```

```
plt.grid(axis='x')
plt.legend()
```