基于线性规划的供应商与转运商选择最优化模型

摘要

生产所需的天然材料供应需要考虑季节性，同时又需要考虑供货量是否满足生产所需。因此企业在选择供应商时需要考虑其供货结构，时间周期，供货量，稳定性等。同时为了减少转运途中的损耗，转运商选择和具体转运方案也是企业需要慎重考虑的方面。本文主要利用**线性规划**的方法，建立最优化数学模型求解各种决策。

针对保障企业生产的供应商重要性评价模型该问题需要从过去五年的订货供货量数据提取出能够量化供应商重要性的指标。经查阅 CIMS-SCM 课题组于 2000 年提出的供应商综合评价指标体系和供应商分类矩阵，提取出了每周的订货量 (Q)、供货量与订货量的比值（定义为稳定性 η)、每个供应商收到订单的周数与总周数（240）的比值（供货柔性 τ)。为了综合三个参数，我们模仿弗鲁姆提出的期望理论：$M = \sum V * E$。得到评价模型为：$S = \sum(lgQ * \eta) * \tau$。根据该模型对 402 个供应商进行评价，具体前五十名名单在后文中。

针对经济性的原材料订购与转运最优策略模型通过**傅里叶变换**进行周期性分析，得到周期为 24 周。再进行按周期平均来作为预期供、订货量数据。我们采用线性规划的思想，以最小化供应商数量为目标函数，约束条件为每周库存量要能满足两周生产所需、选中一个供应商后订购该供应商的所有供货和每周总订购量不能大于转运能力的上限（48000 立方米）。使用 Matlab 编写算法程序，计算得到至少需要 58 家供应商。至于最经济的策略选择，我们同样使用线性规划的方法，目标函数变为了最小化 24 周生产所需的成本。约束条件与求最少供应商时区别在于每周可选择不同的供应商。利用算法得到最优策略订购方案记在附件中。最后我们得到的订购方案总共花费 509762（假设原材料 C 价格为 1），而最小化供应商模型得到的方案则会花费 561822。

针对考虑存储成本的经济性最优策略模型由于考虑到了存储成本，按照单位产能需要对 A、B、C 的优先级进行排序，同时对预计的转运商损耗率数据进行升序排序。随后根据线性规划思想，制定两个目标函数：最大化 A 订购量并最小化 C 的订购量。约束条件与上述经济性订购方案的模型基本一致，此外补充了滞后弥补约束算法。订购 A:192811, 最小的 C：110745, 得到的最终订购方案记录在附件中。

针对产能扩张的最优策略模型由于有限的原材料供应量，我们得到 $A > B > C$ 的优先级排序。沿用线性规划的思想，以最大化产能扩张为目标，约束条件与前几个模型基本相同，需要注意的是库存量需满足两周的扩张后的产能需要。转运方案的制定方法与问题二基本一致。得到最大产能扩张为 8678 立方米。订购方案与转运方案记录在附件中。

关键字： 线性规划　背包问题　傅里叶变换周期性分析　动态规划　评价模型

一、 问题重述

1.1 问题背景

某建筑和装饰板材生产企业需要三种木质纤维和其他植物素纤维材料，分别为A,B,C。不同的供应商会根据各自的供应周期提供各自的供应量，品种。同时，需要确定第三方物流公司（“转运商”）并委托其将供应商每周的原材料供货数量转运到企业仓库，每个转运商有各自的损耗率，同时他们每周的最大运输量是相同的。

利用每种原材料生产分别有其消耗成本，同时，三种原材料价格也不尽相同，总的来说，效率最高的原材料价格最高，效率最低的原材料价格最低。除了满足企业每周产能的原材料需要，为了保证正常生产的需要，企业要求要尽可能保持不少于满足两周产能的原材料库存，因此对于供应商提供的原材料总是全部收购。

1.2 问题要求

- 根据过去五年 402 家供应商的数据，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。
- 参考问题 1，该企业至少选取多少供应商才能满足生产要求。针对这些供应商，为企业制定维拉 24 周最经济的订购方案，并制定损耗最少的转运方案。并对订购方案和转运方案分析。
- 考虑到转运和存储的成本，企业计划尽量多采购 A 类而尽量少采购 C 类材料，以此压缩生产成本。同时希望转运损耗最少。制定新的订购方案和转运方案，分析实施效果。
- 企业如今已有提高产能的潜力，根据现有的供应商和转运商的实际情况，确定企业可以提高多少产能，并指定未来 24 周订购和转运方案。

二、 问题分析

2.1 对问题一的分析

问题一要求建立反映保障企业生产重要性的数学模型来选取供应商，实际上就是对402 家供应商从某些数据维度进行量化分析，建立对这些供应商的评价模型。

根据林勇等在 2000 年[1]提出的供应商分类矩阵，影响力高以及竞争性强被称之为战略性供应商，这意味着影响力和竞争性是评价供应商重要性的指标之一。同时依据附件

[1]林勇, 马士华. 供应链管理环境下供应商的综合评价选择研究 [J]. 物流技术,2000(05):30-32.

一中给出的从第一周到第 240 周 402 家供应商每周的订货量，我们认为越高的订货量代表企业对该供应商的信任度越高，即订货量（Q）也是评价重要性的指量化标之一。

在林勇等提出的供应商综合评价体系中，业务结构和生产能力也是评价供应商影响力的要素，因此我们记每周供货量与订货量的比值记为稳定性（η）作为生产能力的量化表现，而有收到订单周数与总周数（240）之比记为供货柔性（τ）作为业务结构的量化表现。基于此，我们从附件一中找到了用于量化评价供应商重要程度的三个指标。为了从三个指标的维度进行重要性的判断与量化分析，我们借助弗鲁姆提出的期望理论模型[2]进行模拟，将订货量的对数值和稳定性的乘积模拟为效价（Valence），将供货柔性模拟为期望（Expectancy），根据期望公式：$M = \sum V * E$，我们确定供应商重要性评价模型为：$S(Score) = \sum(lgQ * \eta) * \tau$。再根据这个数学模型评价出最重要的五十家供应商。

2.2 对问题二的分析

2.2.1 对于第一小问

根据附件一中各供应商在过去五年里供货的情况，我们发现供应商供货存在周期性，每个周期为 24 周（利用傅里叶变换分析周期性）。于是根据每个周期 24 周，对 240 周的供货量信息进行平均得到每个供应商在一个周期内每一周的供货量数据。经过一系列验证，我们发现单纯每 24 周平均的供货量不足以满足企业的生产需要。出于现实情况，即当某一周订货量为 1 或 0 时，无法简单定义为该供应商在某个周期的这周内没有供应能力，因此我们计算平均值时剔除了所有订货量为 0 或 1 的情况。根据这些以 24 周为一周期的供货量数据，我们根据问题一模型中得到的重要性排名依次向下索引供应商，限制条件为该周供货量加上上一周已有存量大于两周的生产所需。该 * 规划/优化 * 过程我们通过 Matlab 实现。

2.2.2 对于第二小问

问题要求找出最经济的订购方案并据此找出损耗最少的转运方案，我们将问题分为两步走。第一步，找到最经济的订购方案，实际就是找到成本最小的订购方案。首先我们根据题目的信息得到使用原材料 A 和 C 的成本是最低的，使用 B 是最次的选择（从成本最小的角度出发）。于是我们用 Matlab 进行线性规划，目标是使成本最小，将 C 的价格定位基准 1。约束条件为该周供货量加上上一周已有存量大于两周的生产所需，除此之外我们的算法有限选择 AC，知道 AC 选完而没达到生产需求才会继续选择供应 B 的厂商进行订购。

得到订购方案后，我们进行第二步：找到最优的转运方案（损耗最小）。计算转运方案之前，我们用傅里叶变换分析了转运商损耗率的周期性，结果为 24 周为一个周期。

[2]郭惠容. 激励理论综述 [J]. 企业经济,2001(06):32-34.

随后已知订购方案再对这些供货商每周的供货量直接填进转运商是一个十分繁重的计算过程，于是我们对转运商每周损耗率进行排序，得到每周损耗率依次上升的转运商矩阵。这时问题就转换成一个类背包问题，目标从背包价值最大化变成了损耗率最低的问题。要使损耗率最低，即最大化该周损耗率最低的转运商（们）的转运量，再通过 Matlab 进行动态规划进行求解。

2.3 对问题三的分析

解决该题时由于背景不变，可以沿用问题二中经过处理的数据。另外，根据要求，考虑到仓储成本后，经过原料产能比的计算，可以发现三种原材料的仓储成本从大到小依次为 C,B,A，因此在原材料的订购上要优先选择 A 其次 B 最后 C。而在最小化仓储成本的过程中，我们需要保证每周的库存量满足两周的生产所需。依据题目要求，本模型的目标为最小化成本，既要考虑仓储的成本，也考虑转运的耗损。由于不同时期转运商的保存率会影响企业该周的产品接受量，为了最小化转运的耗损，我们借用问题二第二问类似的思想：填满转运率高的转运商后再选择转运率次之的转运商。因此，我们得到了三个目标函数：最大化 A 的订购量、最小化 C 的订购量、最小化转运商的耗损，而为了减少问题的复杂程度，我们模型假定最大化 A 订购量和最小化 C 订购量的优先级高于最小化转运商的耗损，并使用逆推法求得该动态规划问题的最优解。

2.4 对问题四的分析

根据问题一和问题二中数据处理的供货商数据，企业生产的需求实际上基本“榨干”了各个供货商的供货量。但是我们处理的供货商数据方法是按周期平均，总体上是比较保守的估计方法，因为单一供货商的供货结构会使供货量在部分时期剧烈下滑，降低总的供货量。如果考虑产能的扩张，实际上是企业快速发展的前期表现，一定程度上对供货商产生影响：增加企业对于供货商的重要性，从而激励供货商的供货倾向与生产倾向。因此，对于供货数据需要更加乐观的估计。为了得到产能扩大的理论大值，我们对每个供货商进行最乐观的预计：预计下一个周期内的每一周过货量为该供货商在以往多个周期中这一周的最大值。

得到足够的供货量数据后，我们开始对提升产能的订购和转运方案进行制定。要求得能提升多少产能实际上就是要最大化提升的产能，并且带有一系列以上问题已经提及的约束条件。对于提升产能，最优的原材料选择次序为 A>B>C，原因是每一单位的原材料 A 能才来 1.667 单位的产能提升，B 为 1.5152，C 为 1.3889，同时每周的最大转运量是有限且固定的 48000。因此，在规划中，我们算法的遍历将由 A、B、C 的次序遍历。我们利用 Matlab 进行规划求出产能的提升与具体的订购方案。得出订购方案后，我们利用和问题二第二小问中转运方案的制定一样类背包问题的动态规划思想确定最优的转运方案。

三、 模型的假设

- 本文的模型都将根据过去的供货量数据对未来数据进行预测即将供货量作为预测的根据
- 在计算生产成本时，将原材料 C 的成本作为单位 1，因此原材料 A,B 的成本分别是 1.1，1.2
- 问题二第二问与问题四订购方案的求解中预先假定损耗率为百分之一，防止后续计算转运后导致库存量的不足
- 企业对供应商实际提供的原材料总是全部收购
- 一家供应商每周供应的原材料尽可能由一家转运商运输
- 三类原材料运输和仓储的单位费用相同
- 企业只在这 402 家供货商和 8 家转运商中考虑产品订购和转运问题

四、 符号说明

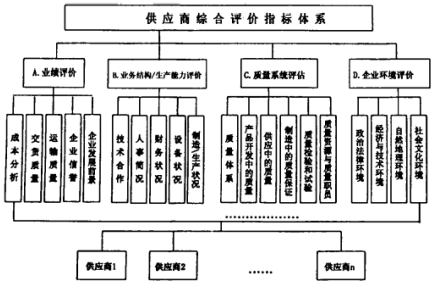
符号	解释
η	稳定性，定义为该供货商该周供货量与订货量的比值
τ	供货柔性，定义为收到订单的周数与总周数的比值
S	供货商在重要性评价模型中的得分
Q(i,j)	过去五年各供货商的订货矩阵
F(i,j)	一个周期内各供货商供货矩阵
n	选中的供货商数量
Y(i,j)	特定周特定供货商是否选中的判断矩阵
D(i,j)	得到的最优订购方案
Tf(i)	原料与产能单位转换向量
P(i,j)	一个周期内各供货商的供货所支持的产能矩阵
T_j	第 j 周生产结束后企业的库存量
V(i)	与供货商供货类别对应的价格向量
R(i,j)	一个周期内各转运商的损耗率数据
G(i,j)	最佳的转运方案
H	扩张的产能
M	转运方案中的转运量
ϕ	每周新增的供应量
ξ_j	转运率（1-损耗率）函数，以转运量为横坐标，根据每周转运商数据决定
X(i，j)	每周 A、B、C 的订货量

五、 模型的建立和求解

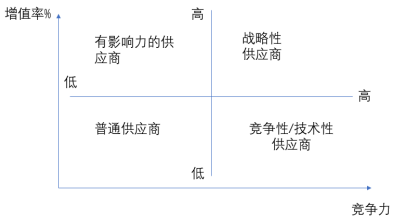
5.1 供应商重要性评价模型

5.1.1 模型的建立

问题一要求旨在借助附件一的数据（402 家供应商在过去五年的供货订货数据）建立一个供应商重要性量化评价模型。在建立模型之前需要先确定量化影响各供应商重要性的指标。通过查阅华中理工大学 CIMS-SCM 课题组调查的数据以及借此林勇教授等提出的供应商综合评价指标体系、供应商分类矩阵，我们确定了三个重要的影响指标：订货量、稳定性和供货柔性。供应商分类矩阵中，高影响力的供应商具有增值率高和竞争性的特征，换言之影响力高的供应商被定义为战略性供应商也就是重要程度高的供应商。附件一的订货量实则便是供应商对企业影响力的量化体现，因此我们将订货量确定为评价模型量化指标之一。不同于订货量，供应商的稳定性反映的是供应商自身的业绩能力评价，符合综合评价指标体系中的业绩评价。同时附件一中表明，大部分企业都不会在过去五年中每一周有供货，也就是说各个供应商有自己的业务结构（供货周期）。



(a) 供应商综合评价指标体系



(b) 供应商分类矩阵

确定了三个具体的量化指标后，我们借助弗鲁姆提出的期望理论（Expectancy theory）进行模拟和改进，建立供应商重要性评价模型。期望理论中有三个重要参数：M，激励（Motivation）；V，效价（Valence）；E，期望（Expectancy）。期望理论认为人们采取某项行动的激励力（M）取决于行动结果的价值评估（效价 V）和预期达成该结果可能性的估计（期望 E）。

$$M = \sum V * E \tag{1}$$

模仿弗鲁姆的期望模型，供应商的重要性是企业对供应商的最终评价，作为期望公式中激励力的延申推演。稳定性的定义是供货量与订货量的比值作为业务能力的量化表现，而由于订货量与稳定性的乘积即是供货量，与期望理论中的效价的定义有相似之处，但是为了剔除订货量过少而不供货从而导致稳定性异常为零的情况（出于成本考虑，现实情况下厂商接到过低的订单不给予供货，但不能说明其稳定性低），我们用订货量的对数与稳定性的乘积 $(lgQ * \eta)$ 充当效价的地位（可以看作计量经济学中的替代变量）。供货柔性为供应商供货周数与总周数（240）的比值，反映的是供应商在过去五年中实际

供货的时间比，也就是供应商供货柔性，为企业生产实际动态需求供应能力的表现，比值越高供应商迎合企业生产需求的动态要求的能力越强。因为供货柔性是以时间的比值呈现与概率的定义十分吻合，因此在模型中充当期望的定位。于是，供应商重要性评价模型建立为：

$$S = \sum (lgQ * \eta) * \tau \tag{2}$$

5.1.2 求解

我们将附件一中的数据导入到 Matlab 中，在进行计算之前，我们对数据进行了预处理：

出现订货量为时，我们将其稳定性定义为 0，是整个乘积项变为 0，避免了对数项出现负无穷的情况。

所有供应商的分数记在矩阵 E(:,3)（见附录）, 分数前五十的供应商 ID 及其分数如下所示：

供应商 ID	分数	供应商 ID	分数	供应商 ID	分数	供应商 ID	分数	供应商 ID	分数
S229	751.4556	S131	652.4346	S365	540.9094	S266	388.2785	S291	170.0716
S361	742.8804	S330	631.4924	S040	506.9118	S007	370.9745	S314	168.9113
S108	688.4394	S356	630.7046	S364	494.6851	S123	364.1605	S150	157.8562
S340	681.0322	S308	630.6828	S367	481.175	S139	300.564	S074	149.1374
S275	677.0031	S194	630.0343	S346	479.8094	S348	287.5384	S037	147.1648
S329	674.5037	S352	607.6047	S294	466.0906	S003	239.0258	S338	140.5964
S282	664.2308	S247	574.93	S055	446.5039	S374	234.5664	S086	135.0851
S151	660.3351	S143	561.0483	S080	444.6124	S140	192.1432	S307	97.18849
S268	656.9453	S031	541.337	S218	442.5676	S114	188.257	S210	94.84393
S306	652.7764	S284	540.9521	S244	436.4456	S189	172.6537	S005	93.03518

5.2 针对问题二基于线性规划的方案选择最优化模型

5.2.1 数据的预处理

企业生产的原材料主要是木质纤维和其他植物素纤维材料，该类天然材料的开采、加工都是具有季节性的，同时储存的方法和环境比较苛刻，因此该类材料的供应一般具有周期性。我们首先对附件一中所有供应商的供货量利用傅里叶变换分析周期性。结果如下图：

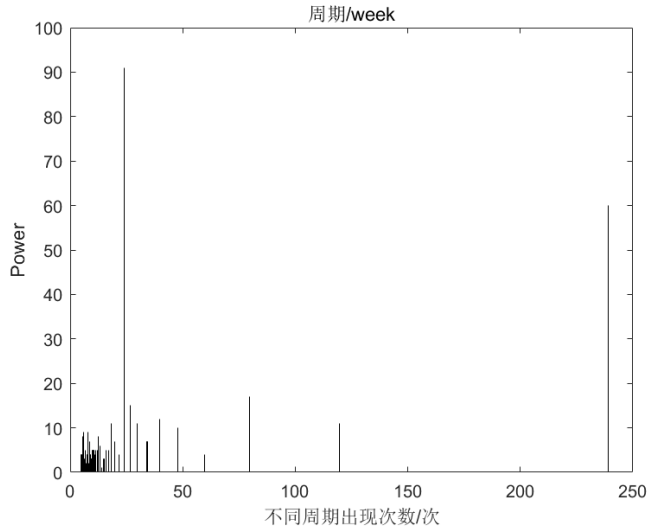


图 **2**　周期分析结果

显然，最大部分的供应商都具有相当明显的周期性，一个周期为 24 周。

同时我们进行时间序列分析，发现供应商供货量数据并不具有趋势性，于是我们对供应商过去五年的供货数据按 24 周为一个周期得到了每个供应商一个周期内每周的供货量平均数据，并将其作为接下来需要制定订购方案的基础。

进行模型建立之前，我们先比对了 24 周生产总共产能为：$28200 * 2 + 28200 * 23 = 707000$ 立方米，而所有供应商的在一个周期内的供应所能支持的产能为 440000 立方米。同时，我们发现简单的对供应商在每个周期内的同一周进行平均来展现其供应能力忽略了订单对其的影响。于是，我们决定剔除订单量为 1 或 0 的情况，因为这种情况下，不少供应商因为成本问题，选择不供应，而这时他的供货量为 0 大大降低了该供应商在这一周的平均供应量。剔除特殊数据后，所有供应商一个周期的总供应量可支持的产能为 803431 平方米，可用于后续优化模型的建立。

5.2.2 针对问题二第一小问的模型建立与求解

目标函数　要求尽量少的供应商数目来满足企业的生产需求，我们将最优策略定义为供应商数目尽可能少且满足生产要求的订购方案，目标函数为：

$$min\ n \tag{3}$$

约束条件　由于选择了一家供应商一个周期内都要向其订购。我们建立一个 01 矩阵 Y 来判断订购方案：$D = F * Y$, 约束条件为判断完一家供应商是否选取后，01 矩阵的该列都只会是 0 或者只能 1：

$$Y(:, j) = 0 | Y(:, j) = 1 \tag{4}$$

由于每家转运商转运能力为 6000 立方米/周，总共有 8 家转运商，即每周能转运的原材料最多为 48000 立方米。因此将、一周内的供应量之和必须小于 48000：

$$\sum D(:,j) \leq 48000 \tag{5}$$

由于原材料与产品并非一对一的关系，而是：

	原材料 A	原材料 B	原材料 C
单位原材料产能	1.667	1.515	1.389
单位原材料价格	1.2	1.1	1

因为每家供应商只供应一种原材料，所以我们引进原料产能转换向量 $Tf(i)$, 再利用 Matlab 的矩阵索引功能对原材料供应矩阵和转换向量对应相乘 $P(i,j) = F(i,j) * Tf(i)$, 得到产能矩阵 P(i,j)。

企业要求尽可能保持不少于两周生产需求的原材料库存量，因此我们创建了一个随时间改变的变量：库存量 T_j, 该量等于上一周结束的库存量加上本周收到的供应量 $\sum P(:,j)$ 大于一周的生产所需，同时定义第一周初始时，也就是第 0 周结束后的库存为零：$T_0 = 0$，因此约束条件：

$$\begin{aligned} T_0 &= 0; \\ T_j = T_{j-1} &+ \sum P(:,j); \\ T &\geq 28200 \end{aligned} \tag{6}$$

优化模型的确定　要求出可以满足产能的最少数量的供应商，我们使用了线性规划的方法，标准型为：

$$min\ n \tag{7}$$

$$\begin{cases} Y(:,j) = 0 | Y(:,j) = 1; \\ \sum D(:,j) \leq 48000 \\ T_0 = 0 \\ T_j = T_{j-1} + \sum P(:,j) \\ T_j \geq 28200 \end{cases} \tag{8}$$

求解与注意事项　第一条约束条件表示的是该周的总供应量不能大于转运商的最大运载量即 6000*8 等于 48000；其次为了满足要求的每周库存必须满足两周的生产需要，增加了第二第三第四条约束条件。

算法中，我们每遍历一个供应商，该供应商会在接下来一个周期都供应。同时为了简化计算，我们将按照问题一中得到的供应商重要性次序从上向下筛选。利用 Matlab 进行线性规划，得到结果为 58 家供应商。

需要注意的是，我们在多次运行该规划模型后，发现有一家供应商（ID 为 S140）的供应十分奇特：240 周里只有十几周有供应，而且这十几周每周的供应量都极为庞大，但由于其稳定性为 0 的周数繁多，所以在重要性排序中排到了靠后的位置。由于我们算法的遍历过程是按照重要性排序进行的，这一现象导致算法结果远大于真实结果（为了满足约束条件，算法会先遍历排名比 S201 高的供应商，但这些供应商大部分供应量都小于 S201，导致了远远偏大的结果）。为了解决这个问题，我们修改算法，默认选上 S201。

此外，第 9、10、11、12 周处于大部分供应商的供应低谷期：这四周即使向所有供应商订货，仍然无法达到满足两周生产的要求的库存，因此我们的算法跳过了这四周。得到结果 58 后，我们反复调试，发现该 58 家供应商与全选所有供应商所供应的量相差很小，大体上满足企业“尽可能保持不少与两周生产需求的原材料库存量。

5.2.3 针对问题二第二小问的模型建立与求解

由于问题要求找到最经济的订购方案并据此找出损耗最少的转运方案，我们将该要求拆分为两部分，第一部分同样通过类似的线性规划算法找到最经济的订购方案，随后根据此订购方案制定最优（损耗最少）的转运方案。

第一步：制定最经济的订购计划 根据原材料之间的价格关系，我们可以计算单位产品的成本（以原材料 C 的价格为单位 1）：

	A	B	C
单位产品消耗	0.6	0.66	0.72
单位原材料价格	1.2	1.1	1
单位产品成本	0.72	0.726	0.72

根据表格，我们发现原材料 A 和 C 的对于生产单位产品来说成本是一样的，都小于原材料 B，因此我们修改第一小问的算法，优先遍历排名高的供应商放到第二优先级，优先遍历供应原材料 A 和 C 的供应商提到第一优先级，直到所有供应 A、C 的供应商该周供应量总和不足以满足拥有两周生产所需库存的要求，才选择原材料 B。

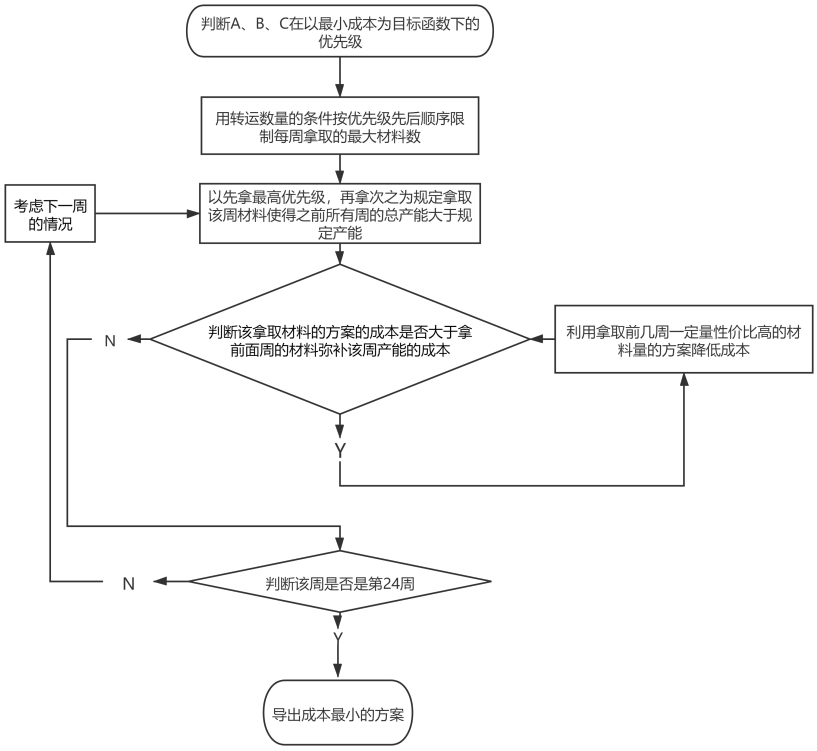


图 **3**　流程图

- **目标函数**

  与第一小问的目标不同，问题要求的是最经济的订购方案实则是成本最小的方案，因此我们的最优策略定义为成本尽可能少的订购方案，成本定义为供应量与各种原材料的价格的乘积（以 C 为单位价格）。定义成本（价格）向量 V(i)，因此目标函数：

$$min\ D(i,j) * V(i)^3 \tag{9}$$

- **约束条件**

  这一小问的约束条件大抵与第一小问的相等，区别在于第一小问对于已选择的供应商会在一个周期内持续订购，但是这一小问可以在不同的周里选择不同的供应商，也就是说 Y 的同一列不再只会是 0 或者只会是 1，而可以在同一列中即存在 0 也存在 1。

- **优化模型的确定**

$$min\ D(i,j) * V(i) \tag{10}$$

$$\begin{cases} \sum D(:,j) \leq 48000 \\ T_0 = 0 \\ T_j = T_{j-1} + \sum P(:,j) \\ T_j \geq 28200 \end{cases} \tag{11}$$

[3]Matlab 中的索引对应位置相乘

- **求解与注意事项**利用 Matlab 程序，按照 A,C,B 的顺序遍历，得到了最经济的订购方案 D(i,j)。经过计算，最经济的订购方案总成本为 509762，而第一问中 58 个供应商全部选上的总成本为 561822，显然大大减少了成本。

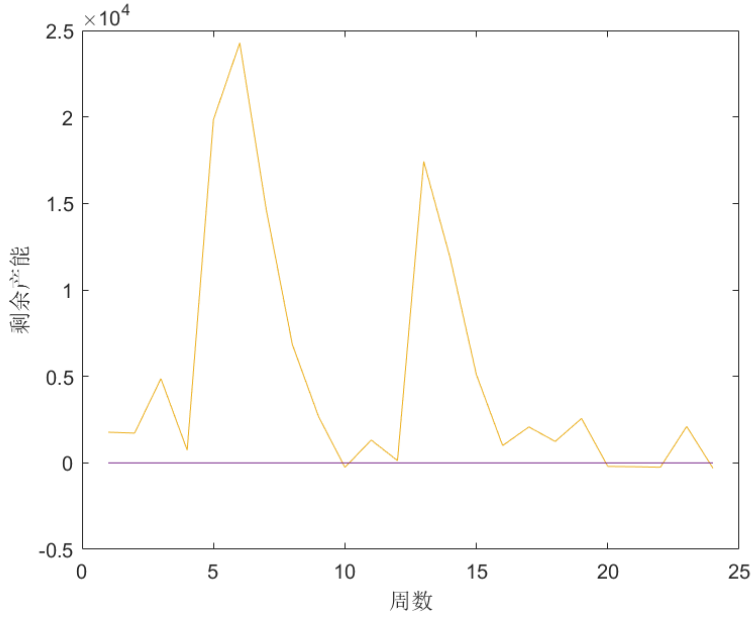


图 **4**　约束条件的满足情况

图 4 表明供货量提供的产能减去生产所需的剩余，表明达成目标函数的同时满足了约束条件。

第二步：转运方案

- **转运商数据的预处理**

  同样利用傅里叶转换的方法对转运商的数据进行周期分析，结果同样具有周期性，为24 周。

  与问题一采取一样的方法取得转运商一个周期内的转运数据（损耗率）R。具体结果见附录。

- **类背包问题**

  我们对转运商每周的损耗率进行排序，得到每周损耗率依次上升的转运商矩阵 R_1。为了最小化损耗率，供应量越大的供应商应选择越好（损耗率越低）的转运商。而当供应量大于 6000 时，就应该把这一供应商放到最优的转运商来保证最小的损耗率。因此我们首先筛选出一次供应大于 6000 的对应的供货商以及周数，将这些元（在矩阵中对应这些供货商和周数）对应位置到最优的转运方案矩阵 F 中。剩下的元对应到最优转运方案矩阵。因为要求一家供货商尽量由一家转运商转运，因此会出现在一家转运商没有达到转运上限时就要选择下一家转运商。这时转换成了一个类背包问题，每一周里，小于背包容量表现为：每一转运商每周最大转运量 6000 的限制下，同时最大化价值表现为：按照优劣次序选择转运商并且最小化转运商商家数量的选择，选择的标准是尽量填满更优的转运商再选择下一位转运商。定义最优转运方案为 G(i,j)。

- **目标函数**作为类背包问题，本来的目标函数是最大化背包中物品的价值，在本模型中，每选择一家转运商进行一次类背包问题的动态规划，因此目标函数为第 j 周中，从第 1 个转运商到第 i 个转运商依次最大化 G(i,j)

$$max\ G(i,j) \tag{12}$$

- **约束条件**首先需要满足的约束条件小于背包容量，在本模型中即为小于总转运量：

$$M \leq 48000 \tag{13}$$

  本模型与单纯的背包模型相比有所创新：模型优先让损耗率低的转运商最大化，所以约束条件为：

$$max\ G(i,j)\ aftermax\ G(i-1,j) \tag{14}$$

- **优化模型的确定**综上所述，该转运方案最优策略选择的线性规划模型的标准型为：

$$max\ M \tag{15}$$

$$\begin{cases} M \leq 48000 \\ max\ G(i,j)\ aftermax\ G(i-1,j) \end{cases} \tag{16}$$

- **求解与注意事项**将算法导入 Matlab 进行计算，得到最优的（损耗率）最低的转运方案，具体结果见附件。

5.3 针对问题三的模型建立与求解

考虑到仓储成本，原材料 A、B、C 的优先级顺序又发生了变化，根据原料产能比能计算出不同原材料之间的相对仓储成本：

	A	B	C
单位产能需要	0.6	0.66	0.72
提供单位产能存储成本（以 A 为单位 1）	1	1.1	1.2

又同时需要考虑到转运承办，我们首先利用问题二中已排序的每周的转运商数据，沿用问题二中制定转运方案的思想：损耗率低的转运商达到转运量上限再选择下一个转运商。为了同时满足仓储成本和转运成本的最小化，我们首先将优先级高的原材料 A 放到排序前的转运商中，其次是 B，最后是 C。

定义上图的转运率曲线函数为 $\xi_j(X(i,j))$，每周的 ξ 都会根据该周的转运商损耗率情况有所不同。

这时候可单独考虑仓储成本最小化，也即最大化 A 的订购和最小化 C 的订购。

5.3.1 数据处理

针对问题三建立的模型需要使用的数据可以直接承用解决问题二时所处理过的数据，因此不再进行额外的数据处理。

5.3.2 目标函数

需要最小化转运成本，但是在最小化仓储成本后进行，最小化仓储成本，目标函数有两个：

$$\begin{aligned} &max\ A \\ &min\ C \end{aligned} \tag{17}$$

5.3.3 约束条件

假设每周新增的供应量为 ϕ_j，同时定义每周的 A、B、C 的订货量矩阵 X(i,j),i 可为 1、2、3，分表代表原材料的种类，j 为 1 到 24，代表一个周期内的各周的情况。根据上图可求得 ϕ_j 的表达式：

$$\phi_j = \frac{\int_0^{X(1,J)} \xi_j d\xi}{0.6} + \frac{\int_{X(1,J)}^{X(1,j)+X(2,J)} \xi_j d\xi}{0.66} + \frac{\int_{X(1,j)+X(2,J)}^{\sum X(:,J)} \xi_j d\xi}{0.72} \tag{18}$$

同时此时的库存量：

$$T_j = T_(j-1) + \phi_j - 28200 \tag{19}$$

因此约束函数为：

$$T_j \geq 28200 \tag{20}$$

此外还需满足订购量不超过最大转运量的要求：

$$\sum X(:,j) \leq 48000 \tag{21}$$

5.3.4 优化模型的确定

综上所述，规划模型的标准型为：

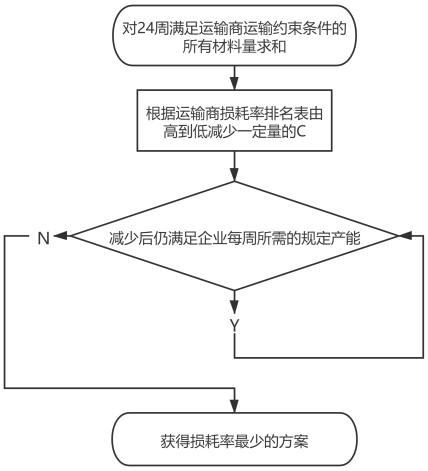
$$\begin{aligned} max\ A &= \sum X(1,j) \\ min\ C &= \sum X(3,j) \end{aligned} \tag{22}$$

$$\begin{cases} \sum X(:,j) \leq 48000 \\ T_j \geq 28200 \end{cases} \tag{23}$$

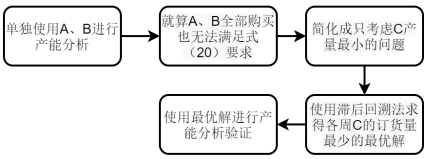
5.3.5 求解与注意事项

进行第一各目标函数最大化时，我们发现将所有 A 供货都订购仍然不满足约束条件公式（20），再将所有原材料 B 都订购后，仍然无法满足公式（20）。因此我们在算法中默认每一周都吧当周所有的原材料 A、B 的供货量全部进行订购，至此 A 已经达到了理论最大值，也即目标函数：$max\ A = \sum X(1,j)$ 已经达成。

进而再进行一个目标函数：$min\ C = \sum X(3,j)$ 的线性规划求解最优策略。然而单纯通过算法遍历所有周数来满足约束条件的方法出现了一个细微的逻辑漏洞：每周都填上欠缺产能所需要的 C，但实际上更优的选择是再损耗率低的周里订购更多的 C，来填补损耗率低的周数，从而使总的 C 的订购量最小。我们通过先订购全部 C, 然后依次按顺序去掉损耗率最高的周数里的 C 的订购量，直到恰好或最接近而又能满足一个周期内总的生产所需（总数满足代表某些不满足公式（20）的周可以从其他有剩余的周数中取出填补）。我们将这一过程成为滞后弥补约束算法。鉴于一个周期内只有 24 周，实际上只需要 24 此筛选，适合人工操作。最后详细的订购和转运方案见附件。



(a) 基于问题三的回溯算法流程图



(b) 问题三流程图

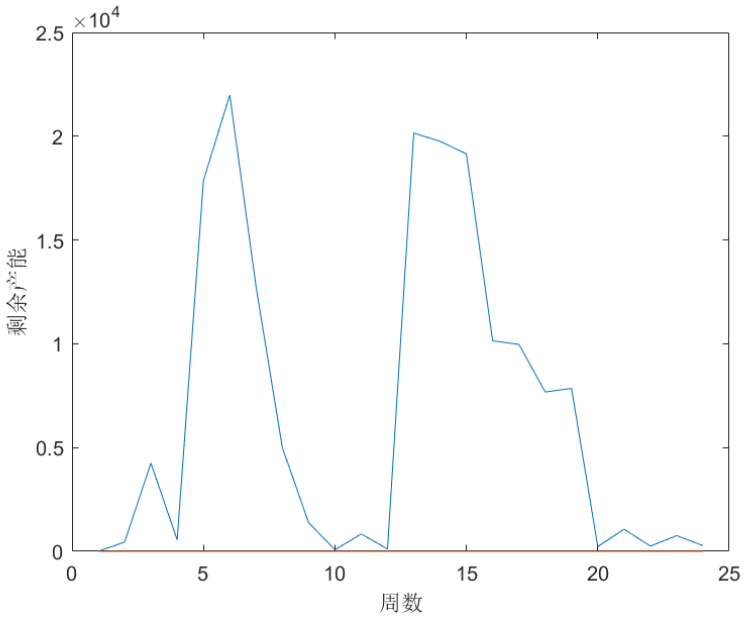


图 **6**　约束条件的满足情况

5.4 针对问题四的模型建立与求解

5.4.1 数据处理

在解决前面的问题时，我们对供货商过去五年的供货数据进行处理，得到了未来一个周期 (24 周) 的估计数据。但是我们使用的方法时按周期平均，是相对来说保守的方法。原因：供应的货物属于木质纤维和其他植物素纤维材料，生产与加工的周期性很强。规模不大的供货商在遇到极端气候时，生产和储存的成本大大提高，为了缩减成本（规模不大导致其无法通过大量供应来弥补成本的增加）而选择不供应进而极大地减少了供货平均值。然而，产能的扩张往往是企业快速扩张、发展的前兆，一定程度商能激发供货商的供货积极性，使供货商在更困难（成本高）的时期也愿意供货。因此，这种情

况下需要对供货数据进行更乐观的估计。于是，我们预计下一个周期内每一周的供货量是过去十个周期里该周的最大供货量。由供货量矩阵 F(i,j) 和原材料产能转化向量 Tf(i) 的对应乘积得到产能矩阵 P(i,j)。

5.4.2 订购计划最优策略选择模型建立与求解

得到合适的供货量数据后，要尽量是增加的产能最大化，要尽量优先选择更多的原材料 A，其次是 B，尽量少订购原材料 C。原因是原材料 A 的边际产量（单位增加对产能的增加）是最大的，B 次之，C 再次：

	A	B	C
单位产量所需	0.6	0.66	0.72
原材料的边际产量	1.667	1.515	1.389

根据遍历的优先级，我们改变供应商矩阵 F 中各个供货商的次序，以满足我们设定的算法运行时的检索顺序。

随后的步骤和问题二第二问的步骤基本类似，可以将该模型的建立和求解定义为线性规划问题。通过 F(i,j) 和判断矩阵 Y(i,j) 的对应相乘可得到最优策略订购方案矩阵 D(i,j)。

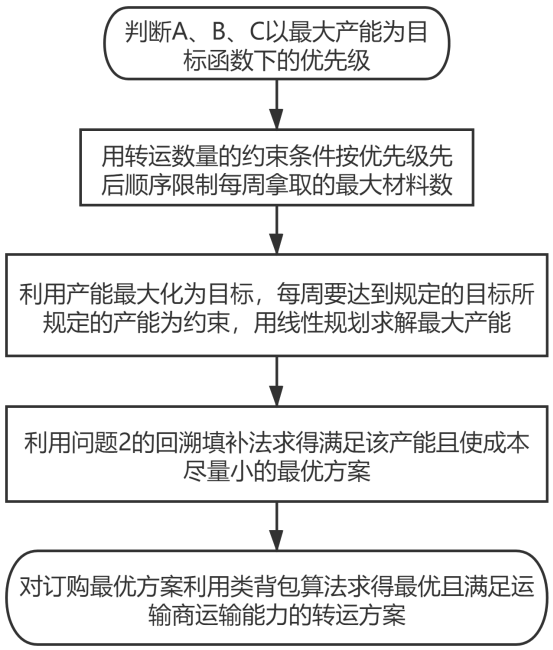


图 **7**　流程图

目标函数 产能扩张达到最大实际上就是最大化产能的问题。我们将新扩张的产能定义为 H，因此目标函数是：

$$max\ H \tag{24}$$

约束条件 每周的最大转运量为 48000 立方米，因此每周的原材料供应量不能大于 48000 立方米：

$$\sum D(i,j) \leq 48000 \tag{25}$$

每周库存量要满足两周生产的需求：

$$T_j \geq 28200 + H \tag{26}$$

优化模型的确定 线性规划的标准型为：

$$max\ H \tag{27}$$

$$\begin{cases} \sum D(:,j) \leq 48000 \\ T_j \geq 28200 + H \end{cases} \tag{28}$$

求解与注意事项 在具体的规划过程中，我们发现了简单的规划算法有逻辑漏洞：当某一周的 A、B 供应之和都不足以满足该周需要供货量，算法会下探选择订购原材料 C，但实际上可以通过这周之前过度订购 A、B 来填补这一周的空缺，能够更大限度地增大产能的提升。A、B 充足时，也可以选择提前订购更多的 A 来弥补该周的空缺。我们将这补充的步骤称为回溯填补法，具体表现为在出现空却的周数，且最近的周数都有比该周使用最低优先级原材料高的原材料的剩余，之前过度订购高优先级的原材料来填补这些周的空缺。后续补充的算法由于限制条件太多，程序代码太冗长低效，而且实际迭代次数不多，遍历周数只有 24 周（一个周期），我们采用更为高效的人工代替程序计算。最终求得增加的茶能最大值为 8678 立方米，具体的订购方案见附件。

5.4.3 制定转运计划的模型建立和求解

根据该订购模型制定转运模型的本质与问题二第二问中制定转运方案的本质相同，可以用相同的类背包问题的动态规划的思想来求解，这里就不作赘述。具体转运方案见附件。

六、 模型的推广与反思

6.1 反思

问题二的第二问、问题三与问题的一些部分都需要人工对数据进行修正。数据量大的时候需要改进算法。

我们的代码普遍性弱、需要具体问题具体分析

6.2 推广

在一定的限制条件下，模型可以精确的解出最优解。

使用的算法具有独创性，在计算量过大的问题中，避免了使用动态规划问题的常规解题思路。

参考文献

[1] 林勇, 马士华. 供应链管理环境下供应商的综合评价选择研究 [J]. 物流技术,2000(05):30-32.

[2] 郭惠容. 激励理论综述 [J]. 企业经济,2001(06):32-34.

附录 A　转运商排行

1	周排名	转运率
2	W012_1	99.97547
3	W008_1	99.97101
4	W022_1	99.97101
5	W010_1	99.95477
6	W016_1	99.95468
7	W002_1	99.94203
8	W021_1	99.94203
9	W022_2	99.90936
10	W021_2	99.90613
11	W015_1	99.89964
12	W009_1	99.88995
13	W008_2	99.88653
14	W023_1	99.88022
15	W007_1	99.82038
16	W004_1	99.79628
17	W007_2	99.72902
18	W009_2	99.72326
19	W002_2	99.69165
20	W003_1	99.64062
21	W010_2	99.54068
22	W024_1	99.53623
23	W019_1	99.49151
24	W012_2	99.4915
25	W014_1	99.48381
26	W021_3	99.42371
27	W011_1	99.40677
28	W015_2	99.40676
29	W018_1	99.37945

图 **8**

附录 B　部分算法代码

```
%%
%问题2.1
%对订货量为0，1的数据进行处理，算出问题二和问题三所用的经处理的周期平均值
for i =1:402
   for j=1:240
      if d(i,j)==0||d(i,j)==1
```

29	W018_1	99.37945
30	W020_1	99.36232
31	W018_2	99.33898
32	W022_3	99.33897
33	W016_2	99.33387
34	W023_2	99.32202
35	W003_2	99.30853
36	W019_2	99.29907
37	W023_3	99.29799
38	W020_2	99.28812
39	W024_2	99.27044
40	W022_4	99.26639
41	W022_5	99.21988
42	W021_4	99.20766
43	W014_2	99.20337
44	W015_3	99.20215
45	W017_1	99.14896
46	W011_2	99.09577
47	W004_2	99.02339
48	W013_1	99
49	W016_3	98.95393
50	W013_2	98.9362
51	W005_1	98.92924
52	W012_3	98.90775
53	W006_1	98.88134
54	W017_2	98.87785
55	W001_1	98.86793
56	W006_2	98.86528
57	W001_2	98.66747
58	W005_2	98.66747

图 **9**

```
            g(i,j)=1000000;
        else
            continue
        end
    end
end
f=zeros(402,24);
for k=1:402
    for i=1:24
        t=0;
        q=0;
```

```
        for e=1:10
            j=i+(e-1)*24
            if g(k,j)~=1000000;
                t=t+1;
                q=q+g(k,j);
            end
            if t~=0
                f(k,i)=q/t;
            else
                f(k,i)=0;
            end
        end
    end
end

 %问题2.2
%计算每周所需要的B类产品的代码
C=zeros(1,24);
t=zeros(14,24);
T=zeros(1,24);
T(1)=10+28200;
for j=2:24
    C(j)=(28200-(T(j-1)+A(j)*0.99-28200))/0.99;
    if j==5
        T(j)=28200-C(j)+26207;
    else

        if C(j)<=0
            T(j)=28200-C(j);
        else
            for i=1:14
                if C(j)-B(i,2*j)>=0
                    C(j)=C(j)-B(i,2*j);
                    t(i,j)=B(i,2*j-1);
                else
                    continue
                end
            end
            T(j)=28200-C(j)+B(14,2*j);

        end
    end
end
%%
%0-1规划类背包算法核心
rongliang=n;% 背包的容量+1
Weight=W(:,4)';
```

```
Value= D(:,4)';
geshu =length(Weight);% 个数
TR=[];%状态转移矩阵
state=[];%状态属性

%1.第一个的判断开头;
for xh=1:n
       if Weight(geshu)<xh
              TR(geshu,xh)=Value(geshu) ;
       else
              TR(geshu,xh)=0;
       end
end

%2.迭代放不放，不放时：F[i,v]=F[i-1,v]；放时：F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i}；.
for pd=geshu-1:-1:1
       for xh=1:n
              if xh<=Weight(pd)
                     TR(pd,xh)=TR(pd+1,xh);
              else
                     if TR(pd+1,xh)>TR(pd+1,xh-Weight(pd))+Value(pd)
                            TR(pd,xh)=TR(pd+1,xh);
                     else
                            TR(pd,xh)=TR(pd+1,xh-Weight(pd))+Value(pd);
                     end
              end
       end
end
TR;

%计算出0-1矩阵

pd=rongliang;
for xh=1:geshu-1
       if TR(xh,pd)==TR(xh+1,pd)
              state(xh)=0;
       else
              state(xh)=1;
              pd=pd-Weight(xh);
       end
end

if TR(geshu,pd)==0
       state(geshu)=0;
else
       state(geshu)=1;
end
```

```
T=state';

%问题2.3
%制造P，令为0的价格很高，让背包无法选零的
for i=1:402
    for j=1:24
        if P(i,j)==0
            D(i,j)=100000;
        else
            D(i,j)=P(i,j);
        end
    end
end

T=zeros(402,24);
for i=1:24 %迭代连续算24次
Rongliang=6001;% 容量
Weight=D(:,i)' ;% 重量.
Value= P(:,i)';% 假设供应量就是价钱
bianhao =length(Weight);
ZYjz=[];%状态转移矩阵
zt=[];%背包里物品的状态

%1.;
for circle=1:6001
        if Weight(bianhao)<circle
                ZYjz(bianhao,circle)=Value(bianhao) ;
        else
                ZYjz(bianhao,circle)=0;
        end
end

%2.下个物品是否放入，不放时：F[i,v]=F[i-1,v]；放时：F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i};
%3.重复2.
for xh=bianhao-1:-1:1
        for circle=1:6001
                if circle<=Weight(xh)
                        ZYjz(xh,circle)=ZYjz(xh+1,circle);
                else
                        if ZYjz(xh+1,circle)>ZYjz(xh+1,circle-Weight(xh))+Value(xh)
                                ZYjz(xh,circle)=ZYjz(xh+1,circle);
                        else
                                ZYjz(xh,circle)=ZYjz(xh+1,circle-Weight(xh))+Value(xh);
                        end
                end
        end
```

```
end
ZYjz;

%4.用0，1代替物品

xh=Rongliang;
for circle=1:bianhao-1
       if ZYjz(circle,xh)==ZYjz(circle+1,xh)
              zt(circle)=0;
       else
              zt(circle)=1;
              xh=xh-Weight(circle);
       end
end

if ZYjz(bianhao,xh)==0
       zt(bianhao)=0;
else
       zt(bianhao)=1;
end

zt;
T(:,i)=zt';
end
%%每次用完T需要创建ch去装载T
%对背包问题进行前面的清洗并进去循环
for i=1:402
   for j=1:24
       if T(i,j)==1;
          P(i,j)=0;
       end
   end
end

for i=1:402
   for j=1:24
       if T(i,j)==1
          D(i,j)=100000;
       end
   end
end
ch1=T;

T=zeros(402,24);
for i=1:24 %迭代连续算24次
Rongliang=6001;% 容量
Weight=D(:,i)' ;% 重量.
```

```
Value= P(:,i)';% 假设供应量就是价钱
bianhao =length(Weight);
ZYjz=[];%状态转移矩阵
zt=[];%背包里物品的状态

%1.;
for circle=1:6001
       if Weight(bianhao)<circle
              ZYjz(bianhao,circle)=Value(bianhao) ;
       else
              ZYjz(bianhao,circle)=0;
       end
end

%2.下个物品是否放入，不放时：F[i,v]=F[i-1,v]；放时：F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i};
%3.重复2.
for xh=bianhao-1:-1:1
       for circle=1:6001
              if circle<=Weight(xh)
                     ZYjz(xh,circle)=ZYjz(xh+1,circle);
              else
                     if ZYjz(xh+1,circle)>ZYjz(xh+1,circle-Weight(xh))+Value(xh)
                            ZYjz(xh,circle)=ZYjz(xh+1,circle);
                     else
                            ZYjz(xh,circle)=ZYjz(xh+1,circle-Weight(xh))+Value(xh);
                     end
              end
       end
end
ZYjz;

%4.用0，1代替物品

xh=Rongliang;
for circle=1:bianhao-1
       if ZYjz(circle,xh)==ZYjz(circle+1,xh)
              zt(circle)=0;
       else
              zt(circle)=1;
              xh=xh-Weight(circle);
       end
end

if ZYjz(bianhao,xh)==0
       zt(bianhao)=0;
else
       zt(bianhao)=1;
```

```
end

zt;
T(:,i)=zt';
end
%%每次用完T需要创建ch去装载T
%对背包问题进行前面的清洗并进去循环
for i=1:402
    for j=1:24
        if T(i,j)==1;
            P(i,j)=0;
        end
    end
end

for i=1:402
    for j=1:24
        if T(i,j)==1
            D(i,j)=100000;
        end
    end
end
ch2=T;

T=zeros(402,24);
for i=1:24 %迭代连续算24次
Rongliang=6001;% 容量
Weight=D(:,i)' ;% 重量.
Value= P(:,i)';% 假设供应量就是价钱
bianhao =length(Weight);
ZYjz=[];%状态转移矩阵
zt=[];%背包里物品的状态

%1.;
for circle=1:6001
        if Weight(bianhao)<circle
                ZYjz(bianhao,circle)=Value(bianhao) ;
        else
                ZYjz(bianhao,circle)=0;
        end
end

%2.下个物品是否放入，不放时：F[i,v]=F[i-1,v]；放时：F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i};
%3.重复2.
for xh=bianhao-1:-1:1
        for circle=1:6001
                if circle<=Weight(xh)
```

```
                    ZYjz(xh,circle)=ZYjz(xh+1,circle);
            else
                    if ZYjz(xh+1,circle)>ZYjz(xh+1,circle-Weight(xh))+Value(xh)
                        ZYjz(xh,circle)=ZYjz(xh+1,circle);
                    else
                        ZYjz(xh,circle)=ZYjz(xh+1,circle-Weight(xh))+Value(xh);
                    end
            end
        end
end
ZYjz;

%4.用0，1代替物品

xh=Rongliang;
for circle=1:bianhao-1
        if ZYjz(circle,xh)==ZYjz(circle+1,xh)
            zt(circle)=0;
        else
            zt(circle)=1;
            xh=xh-Weight(circle);
        end
end

if ZYjz(bianhao,xh)==0
        zt(bianhao)=0;
else
        zt(bianhao)=1;
end

zt;
T(:,i)=zt';
end
%%每次用完T需要创建ch去装载T
%对背包问题进行前面的清洗并进去循环
for i=1:402
    for j=1:24
        if T(i,j)==1;
            P(i,j)=0;
        end
    end
end

for i=1:402
    for j=1:24
        if T(i,j)==1
            D(i,j)=100000;
```

```
        end
    end
end
ch3=T;

T=zeros(402,24);
for i=1:24 %迭代连续算24次
Rongliang=6001;% 容量
Weight=D(:,i)' ;% 重量.
Value= P(:,i)';% 假设供应量就是价钱
bianhao =length(Weight);
ZYjz=[];%状态转移矩阵
zt=[];%背包里物品的状态

%1.;
for circle=1:6001
        if Weight(bianhao)<circle
                ZYjz(bianhao,circle)=Value(bianhao) ;
        else
                ZYjz(bianhao,circle)=0;
        end
end

%2.下个物品是否放入，不放时：F[i,v]=F[i-1,v]；放时：F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i};
%3.重复2.
for xh=bianhao-1:-1:1
        for circle=1:6001
                if circle<=Weight(xh)
                        ZYjz(xh,circle)=ZYjz(xh+1,circle);
                else
                        if ZYjz(xh+1,circle)>ZYjz(xh+1,circle-Weight(xh))+Value(xh)
                                ZYjz(xh,circle)=ZYjz(xh+1,circle);
                        else
                                ZYjz(xh,circle)=ZYjz(xh+1,circle-Weight(xh))+Value(xh);
                        end
                end
        end
end
ZYjz;

%4.用0，1代替物品

xh=Rongliang;
for circle=1:bianhao-1
        if ZYjz(circle,xh)==ZYjz(circle+1,xh)
                zt(circle)=0;
        else
```

```
            zt(circle)=1;
            xh=xh-Weight(circle);
        end
end

if ZYjz(bianhao,xh)==0
        zt(bianhao)=0;
else
        zt(bianhao)=1;
end

zt;
T(:,i)=zt';
end
%%每次用完T需要创建ch去装载T
%对背包问题进行前面的清洗并进去循环
for i=1:402
    for j=1:24
        if T(i,j)==1
            P(i,j)=0;
        end
    end
end

for i=1:402
    for j=1:24
        if T(i,j)==1
            D(i,j)=100000;
        end
    end
end
ch4=T;

%合并2_3的结果
end2=zeros(402,8*24);
V=zeros(402,24);
for j=1:24
    for i =1:402
        if ch1(i,j)==1
            V(i,j)=Q(i,j);
        end
    end
end
for j=1:24
    for i =1:402
        if V(i,j)~=0
            end2(i,(j-1)*8+RANK(1,j))=V(i,j);
```

```
        end
    end
end

V=zeros(402,24);
for j=1:24
    for i =1:402
        if ch2(i,j)==1
            V(i,j)=Q(i,j);
        end
    end
end
for j=1:24
    for i =1:402
        if V(i,j)~=0
            end2(i,(j-1)*8+RANK(2,j))=V(i,j);
        end
    end
end

V=zeros(402,24);
for j=1:24
    for i =1:402
        if ch3(i,j)==1
            V(i,j)=Q(i,j);
        end
    end
end
for j=1:24
    for i =1:402
        if V(i,j)~=0
            end2(i,(j-1)*8+RANK(3,j))=V(i,j);
        end
    end
end

V=zeros(402,24);
for j=1:24
    for i =1:402
        if ch4(i,j)==1
            V(i,j)=Q(i,j);
        end
    end
end
for j=1:24
    for i =1:402
        if V(i,j)~=0
```

```
            end2(i,(j-1)*8+RANK(4,j))=V(i,j);
        end
    end
end

for i=1:402
    for j=1:192
        if hunhe(i,j)~=0
            end2(i,j)=hunhe(i,j);
        end
    end
end

%%
%问题4
%预处理
N=zeros(3,24);
%Y为3*24没有乘上不同类型材料所对应的比值的值
%这一步是求解出在48000约束下的原始最多可以拿多少
P=zeros(3,24);
for i=1:24
    if Y(1,i)<48000
        N(1,i)=Y(1,i);
        P(1,i)=48000-Y(1,i);
        if Y(2,i)<P(1,i)
            N(2,i)=Y(2,i);
            P(2,i)=P(1,i)-Y(2,i);
            if Y(3,i)<P(2,i)
                N(3,i)=Y(3,i);
            else
                N(3,i)=P(2,i);
            end
        else
            N(2,i)=P(1,i);
        end
    else
        N(1,i)=48000;
    end
end
%算出除以每个材料的比值所对应的产能
B=0.99;%平均转运率
T=zeros(3,24);
for i=1:24
    T(1,i)=N(1,i)/0.6*B;
    T(2,i)=N(2,i)/0.66*B;
    T(3,i)=N(3,i)/0.72*B;
end
```

```
%求解最大的产能
for i=1:24
    total1(i)=T(1,i)+T(2,i)+T(3,i);
end
user(1)=total1(1);
for i=2:24
    user(i)=user(i-1)+total1(i);
end

for i=1:24
    user(i)=-user(i);
end

for i=1:24
    U(i)=-1-i;
end
a=U';
a=-a;
b=user';
b=-b;
f=[-1];
[x,fval]=linprog(f,a,b,[],[])
fuzhi=zeros(1,24);
for i=1:24
    iu(i)=T(1,i)+T(2,i);
end
s=36870;
fuzhi(1)=iu(1)-s;
for i=2:24
    if iu(i)+fuzhi(i-1)<s*2
        t(i)=s*2-iu(i)-fuzhi(i-1);
        fuzhi(i)=s;
    else
        fuzhi(i)=fuzhi(i-1)-s+iu(i);
    end
end
%对数据进行预处理分类
T=zeros(402,24);
%A为402*25的矩阵第一列是对应材料类型的数字化结果
for i=1:402
    if A(i,1)==1|A(i,1)==2
        for j=1:24
            T(i,j)=A(i,j+1);
        end
    else
        for j=1:24
            T(i,j)=0;
```

```
        end
    end
end

t=zeros(402,12);
for i=1:402
    if A(i,1)==1
        for j=1:12
            t(i,j)=A(i,j+13);
        end
    else
        for j=1:12
            t(i,j)=0;
        end
    end
end

Conly=zeros(402,24);
for i=1:402
    if S(i,1)==3
        Conly(i,:)=S(i,2:25);
    else
        Conly(i,:)=0;
    end
end

Bonly=zeros(402,24);
for i=1:402
    if S(i,1)==2
        Bonly(i,:)=S(i,2:25);
    else
        Bonly(i,:)=0;
    end
end

Aonly=zeros(402,24);
for i=1:402
    if S(i,1)==1
        Aonly(i,:)=S(i,2:25);
    else
        Aonly(i,:)=0;
    end
end



%排序运行
```

```
C=zeros(402,25);
for i=1:402
    for j=1:402
        if B(i,1)==j
            C(j,:)=B(i,:);
        end
    end
end

C=zeros(402,13);
for i=1:402
    for j=1:402
        if I(i,1)==j
            C(j,:)=I(i,:);
        end
    end
end
%%
%%将2到12行的C数据进行预处理，从而能放入背包问题代码进行计算
W=zeros(402,11);
for i=2:402
    for j=1:11
        if D(i,j)==0
            W(i,j)=100000;
        else
            W(i,j)=D(i,j);
        end
    end
end
for i=1:402
    if ch4(i,1)==1
        now1(i,1)=0;
    end
end

for i=1:402
    if ch4(i,1)==1
        D(i,1)=100000;
    end
end

%前2到12行进行背包问题计算
final=zeros(402,11);
for i=1:11
rongliang=yaode(i)+1;% 容量
Weight=W(:,i)';% 物品的重量，其中0号位置不使用 。
Value= D(:,i)';% 物品对应的价钱，0号位置置为空。
```

```
cd =length(Weight);% n为物品的个数
Tr=[];%定义状态转移矩阵
state=[];%背包里物品的状态

%1.判断第一个物品放或不放;
for pd=1:yaode(i)+1
       if Weight(cd)<pd
              Tr(cd,pd)=Value(cd) ;
       else
              Tr(cd,pd)=0;
       end
end

%2.判断下一个物品是放还是不放; 不放时: F[i,v]=F[i-1,v]; 放时: F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i};
%3.重复2.
for cf=cd-1:-1:1
       for pd=1:yaode(i)+1
              if pd<=Weight(cf)
                     Tr(cf,pd)=Tr(cf+1,pd);
              else
                     if Tr(cf+1,pd)>Tr(cf+1,pd-Weight(cf))+Value(cf)
                            Tr(cf,pd)=Tr(cf+1,pd);
                     else
                            Tr(cf,pd)=Tr(cf+1,pd-Weight(cf))+Value(cf);
                     end
              end
       end
end
Tr;

%4.找出这些物品。

cf=rongliang;
for pd=1:cd-1
       if Tr(pd,cf)==Tr(pd+1,cf)
              state(pd)=0;
       else
              state(pd)=1;
              cf=cf-Weight(pd);
       end
end

if Tr(cd,cf)==0
       state(cd)=0;
else
       state(cd)=1;
end
```

```
final(:,i)=state';
end

%%%对2到12周的C进行循环的计算（可
D=zeros(403,24);
for i=2:403
    for j=1:24
        if P(i,j)==0
            D(i,j)=1000000;
        else
            D(i,j)=P(i,j);
        end
    end
end

for i=2:402
    if now1(i)==0
        D(i)=100000;
    else
        D(i)=now1(i);
    end
end


%%2到12行将C的结果放到AB中

T=zeros(402,4)
for i=1:4
    for j=1:402
        if A(j,i+1)==1
            T(j,i)=A(j,1);
        end
    end
end
jieguo=zeros(402,11)
for i=1:402
    for j=1:11
        if final(i,j)==1
            jieguo(i,j)=kongzhi(i,j);
        else
            jieguo(i,j)=0;
        end
    end
end
for i=1:402
    for j=1:11
```

```
        if jieguo(i,j)~=0
            AB(i,j)=jieguo(i,j);
        end
    end
end
%%
%计算13: 24 B的量
T=zeros(1,12);
B=zeros(1,12);
T(1)=A(1,1)-36880;
for i=2:12
    T(i)=T(i-1)+A(1,i)-36880;
    if T(i)>=0
        B(i)=0;
    else
        B(i)=-T(i);
        T(i)=0
    end
end

%%用背包算法进行计算
final=zeros(403,4);
for i=1:4
rongliang=zhi(i)+1;% 背包的容量
Weight=W1(:,i)';% 物品的重量，其中0号位置不使用 。
Value= D1(:,i)';% 物品对应的价钱，0号位置置为空。
cd =length(Weight);% n为物品的个数
Tr=[];%定义状态转移矩阵
state=[];%背包里物品的状态

%1.判断第一个物品放或不放;
for pd=1:zhi(i)+1
        if Weight(cd)<pd
                Tr(cd,pd)=Value(cd) ;
        else
                Tr(cd,pd)=0;
        end
end

%2.判断下一个物品是放还是不放；不放时：F[i,v]=F[i-1,v]；放时：F[i,v]=max{F[i-1,v],F[i-1,v-C_i]+w_i};
%3.重复2.
for cf=cd-1:-1:1
        for pd=1:zhi(i)+1
                if pd<=Weight(cf)
                        Tr(cf,pd)=Tr(cf+1,pd);
                else
                        if Tr(cf+1,pd)>Tr(cf+1,pd-Weight(cf))+Value(cf)
```

```
                        Tr(cf,pd)=Tr(cf+1,pd);
                    else
                        Tr(cf,pd)=Tr(cf+1,pd-Weight(cf))+Value(cf);
                    end
                end
        end
end
Tr;

%4.找出这些物品。

cf=rongliang;
for pd=1:cd-1
        if Tr(pd,cf)==Tr(pd+1,cf)
                state(pd)=0;
        else
                state(pd)=1;
                cf=cf-Weight(pd);
        end
end

if Tr(cd,cf)==0
        state(cd)=0;
else
        state(cd)=1;
end

final(:,i)=state';
end

%计算16，17，18，24所要的B是哪一些，final是经过背包问题计算的
change1=zeros(402,4);
for i=1:402
    for j=1:4
        if final(i+1,j)~=0
            change1(i,j)=D1(i+1,j);
        end
    end
end
%转运方案的代码直接用第二问的类背包迭代算法
%end4=zeros(402,8*23);
V=zeros(402,23);
for j=1:23
    for i =1:402
        if c1(i,j)==1
            V(i,j)=Q(i,j);
        end
```

```
    end
end
for j=1:23
    for i =1:402
        if V(i,j)~=0
            end4(i,(j-1)*8+RANK(1,j))=V(i,j);
        end
    end
end

V=zeros(402,23);
for j=1:23
    for i =1:402
        if c2(i,j)==1
            V(i,j)=Q(i,j);
        end
    end
end
for j=1:23
    for i =1:402
        if V(i,j)~=0
            end4(i,(j-1)*8+RANK(2,j))=V(i,j);
        end
    end
end

V=zeros(402,23);
for j=1:23
    for i =1:402
        if c3(i,j)==1
            V(i,j)=Q(i,j);
        end
    end
end
for j=1:23
    for i =1:402
        if V(i,j)~=0
            end4(i,(j-1)*8+RANK(3,j))=V(i,j);
        end
    end
end

V=zeros(402,23);
for j=1:23
    for i =1:402
        if c4(i,j)==1
            V(i,j)=Q(i,j);
```

```
        end
    end
end
for j=1:23
    for i =1:402
        if V(i,j)~=0
            end4(i,(j-1)*8+RANK(4,j))=V(i,j);
        end
    end
end

for i=1:402
    for j=1:184
        if dayu6000(i,j)~=0;
            end4(i,j)=dayu6000(i,j);
        end
    end
end
```