生产企业原材料的订购与运输问题

摘要

本文研究生产企业的原材料订购和转运计划。在问题一中，我们建立了衡量供货在数量上的满足程度和供货在时间上的满足程度的两个变量，通过 TOPSIS 熵权法进行综合评价分析确定了最重要的供应商。在问题二中，我们假设供应商的供货能力服从正态分布，建立了最小化供应商数量的随机规划模型。之后在限制供应商数量的情况下建立了双目标随机规划模型来求解最经济的原材料订购方案。在问题三和四中，我们依据实际问题修改了随机规划模型的约束条件和目标函数，求解得到了在最小化原料总体积和最大化产能的情况下的具体转运方案。

在问题一中，我们考虑到相对较优秀的供货商在供货时的总供货量和供货的时效性能够较好的满足供货需求，用总供货数与总订货数的比值和供货量满足需求的周数与订货周数的比值来分别衡量供货在数量上的满足程度和供货在时间上的满足程度。结合上面的两个满足度以及各个供货商具体的供货量三个供货特征，我们使用 **TOPSIS 熵权法**这一综合评价方法来对 402 家供应商的供货特征进行量化分析，确定了各个最重要的供应商的具体情况，求解得到的最重要的供应商是 S229。

在问题二中，我们首先建立了以最小化供应商数量为目标的数学规划模型。我们假设供应商的供货能力服从正态分布，从而建立了一个**随机规划**模型。在考虑运输损耗的前提下，我们依据最小化总成本和最小化运输损耗两个目标函数建立随机规划模型。为了求解两个随机规划模型，我们采用**样本平均近似法** (SAA)，把随机规划问题转化为多个线性规划问题进行求解。得到最小的供应商数量为 11。对于最小化总成本的双目标优化问题，针对每个优化目标我们采用**分层序列法**，优先求解总成本最小的目标函数，之后考虑损失最小的目标函数。求解得到按照损耗最少的转运方案运输时的平均采购成本为 20403，平均订单总产能为 28282，平均耗损总产能为 82。

在问题三中，为了减小转运及仓储的成本，我们应当尽量选择 A 材料而避免选择 C 材料，考虑到这一目标，我们相对问题二增加了最大化 A 材料订购量和最小化 C 原料订货量的目标函数，建立了随机规划模型。我们可以通过 SAA 法把随机规划转化为多个线性规划模型进行求解。得到的最小的平均采购成本为 20450，损耗率为 0.411%，此时原料 A 的周订货量的数学期望为 11521, 原料 C 的周订货量的数学期望为 885。

在问题四中，企业希望在当前供货情况的基础上最大化产能，因而我们的目标函数需要修改为最大化企业的总产能和最小化转运过程的原料损耗，由于描述供货商供货能力的变量是一个随机变量，问题四的优化模型也可以表示为随机规划模型。我们可以通过与问题三和四相同的方法求解随机规划模型。得到在当前供货情况下企业的最大产能为 45159 立方米。

关键字： TOPSIS 法　随机规划　分层序列法　样本平均近似法

一、 问题重述

某板材生产企业每年按 48 周安排生产，需要提前制定 24 周的原材料订购和转运计划。计划的制定过程包括按照产能需求选择供应商，确定对应每家供应商的订货量，以及确定合适的转运商把供应商的供货转运到企业。

企业的每周的产能为 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。但是生产和运输效率较高的 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%。在供应这些原材料时供应商不能保证严格按订货量供货, 实际供货量可能多于或少于订货量，因而企业会全部收购供应商实际提供的原材料。

在转运收购的原料的过程中，原材料会发生一定程度上的损耗，因而企业实际的“接收量”要小于供应商实际的供货量。在问题中，我们需要考虑近 5 年来 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，逐步建立合适的数学模型，来给出最佳的订购和转运计划。

在问题一中，我们首先需要对 402 家供应商的供货特征进行量化分析，建立反映供货商保障企业生产重要性的数学模型，并在此基础上确定 50 家最重要的供应商。

在问题二中，首先我们需要在满足生产的需求的前提下最小化供应商的数量。之后针对使得供应商数量最少的供应商组合，为该企业制定未来 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。最后对订购方案和转运方案的实施效果进行分析。

在问题三中，我们考虑到尽量多地采购 A 类并且尽量少地采购 C 类原材料，可以减少转运及仓储的成本。在希望生产成本和转运损耗率最小的前提下，制定新的订购方案及转运方案，并分析方案的实施效果。

在问题四中，我们假设企业通过技术改造已具备了提高产能的潜力，首先我们需要根据现有原材料的供应商和转运商的实际情况，确定该企业每周的最大产能，并给出最大化产能情况下未来 24 周内的订购和转运方案。

二、 问题分析

在问题一中, 考虑到大多数供货商的供货量数据包含较多的代表未供货的数据以及部分供货量明显超过均值的数据, 直接对附件一进行分析不能够较好的反映供货商的具体供货情况。考虑到相对较优秀的供货商在供货时的总供货量能够较好的满足供货需求，且供货商的供货时间应当能及时匹配当周的订货需求。我们可以使用总供货数与总订货数的比值和供货量满足需求的周数与订货周数的比值来分别衡量供货在数量上的

满足程度和供货在时间上的满足程度。我们可以结合上面的两个满足度以及具体的供货量三个供货特征，使用 TOPSIS 熵权法这一综合评价方法来对 402 家供应商的供货特征进行量化分析，从而确定了最重要的供应商的具体情况。

在问题二中，我们首先需要建立以最小化供应商数量为目标的数学规划模型。我们假设供应商的供货能力服从正态分布，从而建立了一个随机规划模型。在考虑运输损耗的前提下，模型的约束条件包括总供货量必须满足需求，单个转运商有运输的最大上限等。之后我们可以采用 SAA 法，把随机规划问题转化为线性规划问题进行求解，得到最小的供应商数量。

在确定了可行的最小供应商数目后，我们需要针对这些供应商制定最经济的原料订购方案和损耗最少的转运方案。相对问题二的第一部分，我们建立了公司采购总成本最小和运输损失最小两个目标函数，可以通过分层序列法和 SAA 法对上述的多目标随机规划问题进行求解，从而得到未来 24 周内每周最经济的原材料订购与运输方案。

在问题三中，由于问题要求尽量选择单位产能对应体积较小的 A 材料，我们相对问题二增加了最小化原料总体积的目标函数。由于供货商的供货能力和转运商的损耗情况是随机的，问题三的模型同样是一个随机规划模型，我们可以通过 SAA 法把随机规划转化为多个线性规划模型进行求解。

在问题四中，企业希望在当前供货情况的基础上最大化产能，因而我们的目标函数需要修改为最大化企业的总产能和最小化转运过程的原料损耗，由于在问题四中企业产能被修改为了模型的约束条件，问题四的优化模型的约束条件和问题二也有一定差别。我们可以通过与问题三和四相似的方法求解随机规划模型，从而确定企业的最大产能并给出未来的订购和转运方案。

三、 模型假设及符号说明

3.1 模型假设

1. 认为供货商的平均供货能力服从正态分布
2. 企业的每周产能是固定的，不受到外界因素的影响
3. 认为转运商的转运损耗情况服从正态分布

3.2 符号说明

符号	含义
K	所有材料组成的集合
I	所有供应商组成的集合
J	所有转运商组成的集合
P	每周的企业产能
d_{ij}	近 5 年内企业向第 i 家供应商在第 j 周的订货量
g_{ij}	近 5 年第 i 家供应商第 j 周的供货量
w_j	转运商 j 的损耗率
G	第 i 家供应商的周最大供货量
b_i	第 i 家供应商提供的原材料种类
c_i	每立方米产品消耗的第 i 种原材料的量

四、 问题一

4.1 模型准备

某个供应商的供货情况与需求情况如下图:

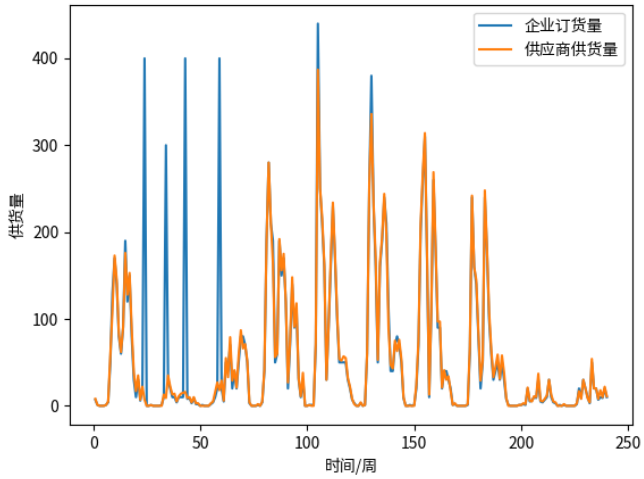


图 1　供货商的供货与订货情况

分析多数供应商的供货情况可知，大多数供货商的供货量数据包含代表未供货或少量供货的数据，以及部分在供货量明显高于平均供货量水平的峰值部分，这代表着在多数情况下，供货商的供货量不能反应具体的供货商的供货能力。因而直接对附件一的供货数据进行分析不能够较好的得到供货商的具体供货情况。我们可以依据供应商的供货情况建立能够表现供应商的供货特征的变量，并利用这些变量来代替供应商的具体供货情况进行分析。

我们希望供货商能够及时，足量的供货。因而我们可以把供货商的供货情况分为在供货在数量上的满足情况和在供货在时间上的满足情况两部分。我们可以用总供货数与总订货数的比值来反映在供货在数量上的满足情况，即：

$$S_a = \frac{\sum_{j=1}^{240} g_{ij}}{\sum_{j=1}^{240} d_{ij}} \tag{1}$$

在讨论供货在时间上的满足情况时，我们可以把供应商在 240 周内接到订单的周数视为供应商的订货周数。由于不同时间由不同转运商转运的损失情况不同，且转运的损耗率的均值约为 1%，具体的损耗量相比于供货量较小。为简化问题，在分析供货情况时可以排除转运损耗带来的影响。在这种情况下，如果周内的供货大于订货，可以认为该周的订单需求被充分满足。

在确定了总的供货周数和满足需求的供货周数后，我们可以用供货量满足需求 ($g_{ij} > d_{ij}$) 的周数 g_w 与订货周数 d_w 的比值来衡量供货在时间上的满足情况，即：

$$S_t = \frac{g_w}{d_w} \tag{2}$$

4.2 模型建立

在利用公式1和2中的比值描述了供货在时间和数量上的满足情况后，我们可以用反映供货在时间和数量上的满足情况的两个比值，以及各个供应商具体的供货量来描述样本总体的供货情况。

为了在合理的考虑到三种因素对保障企业生产重要性的影响情况，我们可以使用 TOPSIS 熵权法 [1] 这一综合评价方法来分析具有不同的上面的三种供货特征的各个供应商在保障企业生产方面的重要性。

4.2.1 熵权法

为了使预测更为客观，我们可以采用熵权法来确定供货在时间和数量上的满足情况以及各个供货商的具体总供货量三个因素。熵权法是一种根据指标变异性的大小来确定客观权重的赋权方法。方法认为一个信息熵较小的指标，指标值的变化程度较大，提供的信息较多应该赋予较大的权重。

设 X_1, X_2, X_3 分别代表供货量、供货在时间上的满足情况、供货在时间上的满足情况三个属性，n 代表供货商的总数。首先，我们需要对 X_1, X_2, X_3 通过模一化操作分别进行标准化：

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}} \tag{3}$$

标准化后得到分别对应的标准化矩阵 $R_k = (r_{ij})_{n\times 3}$。

根据标准化矩阵，利用信息熵法求解每个子集对应的属性权重，首先求出各个决定因素关于属性 X_j 的熵：

$$E_j = -\frac{1}{\ln n}\sum_{i=1}^{n} r_{ij}\ln r_{ij}, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, 3 \tag{4}$$

再求出各供应商在属性 X_j 上的区分度：

$$F_j = 1 - E_j, 0 \leqslant F_j \leqslant 1 \tag{5}$$

对各供货商在每个指标 X_j 上的区分度做归一化处理，可以得到各个供货商在评价指标 X_j 上的权重

$$w_j = \frac{F_j}{\sum_{j=1}^{3} F_j} \quad i = 1, 2, \cdots, n, \quad j = 1, 2, 3 \tag{6}$$

4.2.2 TOPSIS 方法

TOPSIS 法是一种能充分利用原始数据的信息，反映各评价方案之间的差距的综合评价方法。这一方法的基本思想是先根据已有数据构造理想最优解和最劣解。通过评估各个方案距离理想最优解和最劣解的综合距离来实现对各个方案的评价。

在我们的具体求解中，可以依据在熵权法的求解过程中得到的权重向量 W 及标准化矩阵 R 确定最优值向量 Z^+ 和最劣值向量 Z^-，通过计算各个供货商的供货在时间上的满足度、供货在空间上的满足度、供货总量三个属性与最优和最劣值向量的距离实现对各家供货商重要性的综合评价。

首先，我们可以通过熵权法的求解过程得到权重向量 W 及标准化矩阵 R，并把两个矩阵相乘得到加权后的规范化矩阵 Z：

$$Z = (r_{ij}w_j)_{n\times m} \tag{7}$$

我们可以选取 Z 中各项指标最优值和最劣值分别构成最优值向量 Z^+ 和最劣值向

量 Z^- 用做参考。样本与最优值和最劣值的距离为:

$$\begin{aligned} D_i^+ &= \sqrt{\sum_{j=1}^{3}(z_{ij}-z_j^+)^2} \\ D_i^- &= \sqrt{\sum_{j=1}^{3}(z_{ij}-z_j^-)^2} \end{aligned} \tag{8}$$

我们可以通过计算评价对象和最优方案的相对接近度来判断给出各个对象与最优水平间的关系。从而给出各个供应商在保障企业生产方面的重要性情况。

4.3 模型求解

在进行求解前，我们首先按照4.1部分的预处理方法，依据供货商的实际供货情况确定供货在数量上的满足情况和在供货在时间上的满足情况，并且把供货在两方面的满足情况与各个供货商的供货总量结合，用于之后的综合评价。

我们首先使用熵权法确定三个指标对应的权重。根据各家供货商的数据求解方程3、4、6。得到总供货量，周满足度和件满足度三个指标的权重为：0.785560：0.095868：0.118572。

在确定了标准化后的供货数据以及各个供货指标对应的权重后，我们通过 TOPSIS 方法得到如下表的 50 家最重要供应商以及它们的排序情况，供应商的具体得分和类型请见附录。

排序	供应商	排序	供应商	排序	供应商	排序	供应商	排序	供应商
1	S229	11	S131	21	S395	31	S169	41	S175
2	S361	12	S308	22	S307	32	S221	42	S237
3	S140	13	S330	23	S247	33	S040	43	S346
4	S108	14	S268	24	S284	34	S174	44	S324
5	S282	15	S356	25	S201	35	S266	45	S080
6	S151	16	S306	26	S031	36	S342	46	S294
7	S275	17	S352	27	S365	37	S367	47	S141
8	S329	18	S194	28	S374	38	S218	48	S005
9	S340	19	S348	29	S364	39	S037	49	S007
10	S139	20	S143	30	S178	40	S338	50	S092

五、 问题二

5.1 模型建立

在问题二中，我们首先需要在满足生产的需求下最小化供应商数量。之后我们需要针对这些供应商，为该企业制定未来 24 周每周最经济的原材料订购方案，并制定损耗最少的转运方案。

为了方便之后的叙述，我们做出以下定义。1. 最小转运消耗产能：把转运消耗的原料体积折合为对应的产品产能 2. 假设单位体积的原料 C 的价格为 1

5.1.1 最小化供应商数量的随机规划模型

为了最小化供应商数量，我们需要首先确定各个供应商的供货能力，并且在题目给出的约束条件的限制下尽量选择供货能力较强的供应商，从而使供应商总数最小化。

考虑到在多数情况下，供应商的往往处于供货量为 0 或供货量很小的状态（如图1），但也有部分情况下供货商供货量明显高于平均水平，处于不可持续供货的峰值，这两类情况下的供货数据都无法充分的反映供应商的供货能力。

我们需要分析在供货周内供应商的供货量的分布情况。对于各个供应商，由于在去除含 0 情况后对应各个供货商的供货周数不同。我们可以对去除含 0 项后各个供应商在不同的供货周内的供货量进行随机抽样，并且通过 SW 检验 [3] 的方法来对样本的分布情况进行检验。

在取 5 个最大值时，对供货量不同的各个供货商的供货量经过 SW 检验得到的平均 p 值为 0.1405>0.05，没有充分的证据表明多数供货商的供货能力不服从正态分布，不能否定样本服从正态分布的假设。我们可以令 G_i 表示第 i 家供应商的周最大供货量，根据上面的统计学分析，我们可以假设 G_i 服从正态分布，分布的均值为 E($\hat{g}_i$)，方差为 Var($\hat{g}_i$)，其中的 $\hat{g}_i$ 代表当 $d_{ij} \neq 0$（有订货）时 g_{ij} 对应的序列。

此外，考虑到在数据中同种原料有多家供应商供应，由于厂商的总产能有限，在选取了某一家原料商的原料后可能会减少对与该原料商供应相同类型原料的厂商的供货，而增加与原料商供应不同类型原料的厂商的供货，从而可能导致各个样本之间存在一定程度的相关性，我们通过求解各个样本间的相关系数来判断样本间的相关性，并且根据相关系数的取值采用合理的正态分布模型描述各个供应商间的相关情况。

我们求解了供货商 1 与其他各个供货商的相关系数，发现相关系数的均值为 0.0103，相关系数的绝对值较小，说明各个原料商间的相关性很不明显。因而采用单变量的正态分布就可以较好的反映各个供货商的供货能力。

在通过随机变量的分布确定了供货商的供货量的数学表示后，我们可以建立**随机规划**模型来表述最小化供应商数量的优化问题。假设 I_1 为被选中的供应商的集合，我们

建立下列决策变量：

$$s_i = \begin{cases} 1 & i \in I_1 \\ \\ 0 & else \end{cases} \tag{9}$$

参考论文 [2]，我们可以使用样本平均近似法（SAA）生成 K 个具有代表性的场景，其中每种场景出现的可能性为 p_k，且 $\sum_{k\in K} p_k = 1$。

在使用 SAA 法求解随机规划模型时，样本的目标函数可以表示为各个情况下样本的目标函数的加权平均，因而在我们的最小化供货商数量的优化模型中，目标函数可表示为：

$$\min \sum_{k\in K} \sum_{i\in I} p_k s_i^k \tag{10}$$

在排除损耗因素影响后，用 y_{ij} 表示转运商 j 运送供应商 i 的货量，在各家供货商的总供货量大于企业的产能需求时，有在考虑运输损耗的情况下，企业订货量满足企业生产需求的约束条件：

$$\sum_{i\in I} \frac{s_i^k G_i^k - \sum_{j\in J} y_{ij}^k w_j^k}{c_{b_i}} \geq P \forall k \in K \tag{11}$$

否则，各家供货商的总供货量小于企业的产能需求时，企业的订货量等于各家供应商的总供货量，即：

$$s_i^k = 1 \quad \forall i \in I; \forall k \in K \tag{12}$$

上述约束条件可以合并如下，即：

$$\begin{cases} \sum_{i\in I} \frac{s_i^k G_i^k - \sum_{j\in J} y_{ij}^k w_j^k}{c_{b_i}} \geq P \quad \text{if} \quad \sum_{i\in I} \frac{G_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} > P \quad \forall k \in K \\ \\ s_i^k = 1 \quad \text{else} \quad \forall i \in I; \forall k \in K \end{cases} \tag{13}$$

各家转运商转运的来自某个订单的总原料量应当与企业在这个订单的订货量相等，即：

$$\forall i \in I; \forall k \in K \quad \sum_{j\in J} y_{ij}^k = s_i^k G_i^k \tag{14}$$

此外，在转运过程中，单个转运商运输的最大上限为 6000：

$$\forall j \in J; \forall k \in K \quad \sum_{i\in I} y_{ij}^k \leq 6000 \tag{15}$$

在考虑了转运过程中的可能的损耗因素后，最小化供应商数量的随机规划模型可以

整理如下：

$$
\begin{aligned}
&\min \sum_{k\in K}\sum_{i\in I} p_k s_i^k \\
&\text{s.t.} \quad s_i = \begin{cases} 1 & i \in I_1 \\ 0 & else \end{cases} \\
&\begin{cases} \sum_{i\in I} \frac{s_i^k G_i^k - \sum_{j\in J} y_{ij}^k w_j^k}{c_{b_i}} \geq P \quad \text{if} \quad \sum_{i\in I} \frac{G_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} > P \quad \forall k \in K \\ s_i^k = 1 \quad \text{else} \quad \forall i \in I; \forall k \in K \end{cases} \\
&\sum_{j\in J} y_{ij}^k = s_i^k G_i^k \quad \forall i \in I; \forall k \in K \\
&\sum_{i\in I} y_{ij}^k \leq 6000 \quad \forall j \in J; \forall k \in K
\end{aligned} \tag{16}
$$

5.1.2 最小化总成本的随机规划模型

在确定了可行的最小供应商数目后，我们需要针对这些供应商制定最经济的原料订购方案和损耗最少的转运方案。

在综合考虑货物的成本以及货物的转运成本的优化问题中，我们需要最小化总体的订货成本和运输过程中的损耗成本。考虑到描述供货商最大供货能力的变量 G 是一个随机变量，因而我们的优化模型是一个随机规划模型。我们在最小化总成本的随机规划模型中可以使用样本平均近似法（SAA）生成 K 个具有代表性的场景，从而把随机规划问题转化为普通的线性规划问题。其中每种场景出现的可能性为 p_k，且 $\sum_{k\in K} p_k = 1$。用 I_1 表示使得供应商数量最少时对应的供应商集合，x_{it} 表示第 t 周企业向供应商 i 的订货量，y_{ijt} 表示第 t 周转运商 j 运送的来自供应商 i 的原料量。w_{jt} 表示第 t 周转运商 j 的耗损率。我们的优化目标可以用一个双目标规划模型表示。

规划模型的目标函数表述如下：

$$
\min \sum_{k\in K}\sum_{i\in I_1} p_k x_{it}^k u_{b_i} \tag{17}
$$

$$
\min \sum_{k\in K}\sum_{i\in I_1}\sum_{j\in J} p_k y_{ijt}^k w_{jt}^k u_{b_i} \tag{18}
$$

在公式17中，公式 $x_{it}^k u_{b_i}$ 代表了在情形 k 下公司从供应商 i 处采购的采购成本，公式17整体表示了使得货物总价格的数学期望最小的目标函数，而公式18代表使得运输损耗最小的目标函数。

考虑到货物的损失量相对货物的总价值较小（约为货物总价值的 1-2%），且题目中希望优先确定原料的订购方案，并根据订购方案此制定损耗最少的转运方案。我们可以采取分层序列法进行求解，优先考虑使得原料订购成本最小的目标函数17。

在实际生产中，我们的订购方案受到“订购与转运方案必须能满企业的”在最小化总成本的模型中，由于我们考虑到了原料的运输损耗因素，所以我们在约束条件中应当加入对损耗的考虑。

关于优化模型的约束条件，由于我们需要获取能满足企业的生产需求的订购与运输方案。因而一个约束条件可表示为在计算由于配送导致的损耗后，各个供货商供货的材料能够满足每周产能。若总供应量不能满足产能要求时，订货量应等于总供应量。即：

$$\begin{cases} \sum_{i\in I}\frac{x_{it}^k-\sum_{j\in J}y_{ijt}^k w_{jt}^k}{c_{b_i}} \geq P \quad \text{if} \quad \sum_{i\in I}\frac{G_{it}^k-\sum_{j\in J}y_{ijt}^k w_{jt}^k}{c_{b_i}} > P \quad \forall t\in T;\forall k\in K \\ x_{it}^k = G_{it}^k \quad \text{else} \quad \forall i\in I;\forall t\in T;\forall k\in K \end{cases} \tag{19}$$

此外，各个供应商每周的供应能力是有限的，因而在理想情况下企业向供货商的订货量不应当超过供货商的总供应能力。这一约束可以表示为：

$$x_{it}^k \leq G_{it}^k \quad \forall i\in I_1;\forall t\in T;\forall k\in K \tag{20}$$

在关于原料运输的约束条件中，各家转运商转运的来自某个订单的总原料量应当与企业的订货量相等，即：

$$\sum_{j\in J} y_{ijt}^k = x_{it}^k \quad \forall i\in I_1;\forall t\in T;\forall k\in K \tag{21}$$

此外，由于每家转运商的运输能力为 6000 立方米/周，我们应当限制每周内各家供应商的最大转运量

$$\sum_{i\in I} y_{ijt}^k \leq 6000 \quad \forall j\in J;\forall t\in T;\forall k\in K \tag{22}$$

由于我们在选择供应商时增加了在集合 I_1 中选取的隐含条件，因而我们得到的订购方案和转运方案在是在使得供应商总数最少的供应商集合中选择的。

最小化总成本的随机规划模型可以整理如下：

$$\begin{aligned} &\min \sum_{k\in K}\sum_{i\in I_1} p_k x_{it}^k u_{b_i} \\ &\min \sum_{k\in K}\sum_{i\in I_1}\sum_{j\in J} p_k y_{ijt}^k w_{jt}^k u_{b_i} \\ &\text{s.t.} \quad \begin{cases} \sum_{i\in I}\frac{x_{it}^k-\sum_{j\in J}y_{ijt}^k w_{jt}^k}{c_{b_i}} \geq P \quad \text{if} \quad \sum_{i\in I}\frac{G_{it}^k-\sum_{j\in J}y_{ijt}^k w_{jt}^k}{c_{b_i}} > P \quad \forall t\in T;\forall k\in K \\ x_{it}^k = G_{it}^k \quad \text{else} \quad \forall i\in I;\forall t\in T;\forall k\in K \end{cases} \\ &x_{it}^k \leq G_{it}^k \quad \forall i\in I_1;\forall t\in T;\forall k\in K \\ &\sum_{j\in J} y_{ijt}^k = x_{it}^k \quad \forall i\in I_1;\forall t\in T;\forall k\in K \\ &\sum_{i\in I} y_{ijt}^k \leq 6000 \quad \forall j\in J;\forall t\in T;\forall k\in K \end{aligned} \tag{23}$$

5.2 模型求解

在确定两个衡量货物总价格的约束条件和衡量损失总价值的约束条件的影响情况时，我们可以采用分层序列法和样本平均近似法 (SAA)[2] 求解上述的双目标随机规划模型。分层序列法是一种对规划模型中的各个目标按其重要性分类并求解的方法，在我们的问题情境中，由于货物的损失量相对货物的总价值较小（约为货物总价值的 1-2%），因而损失总价值相对货物总价值对总成本的影响较小，此外题目中也希望我们优先确定原料的订购方案，并据此制定损耗最少的转运方案。依据分层序列法，我们可以优先求解使货物的总价值最小的目标函数，之后再考虑运输损耗因素的影响。

SAA 法是样本平均近似法的简称，我们可以采用这一方法来求解我们的随机规划模型。对于随机规划问题

$$\min f(x,\theta)$$

其中 x 是求解变量，θ 是随机变量。SAA 方法用抽样的方法将随机变量用样本表示，从而将随机规划问题转化为多个确定的线性规划问题。假设服从随机变量 θ 的分布的样本为 $\theta_1,\theta_2,\cdots,\theta_K$，原问题可以转化为下面的一些线性规划问题：

$$\min \frac{1}{K}\sum_{i=1}^{K} f(x,\theta_i) \tag{24}$$

我们取 $K=1000$，根据 SAA 方法对最小化供货商数目的随机规划模型进行求解。排除掉不能满足每周产能的情况后，剩余 961 组可行解。最小供货商数量的频次分布情况如图2：

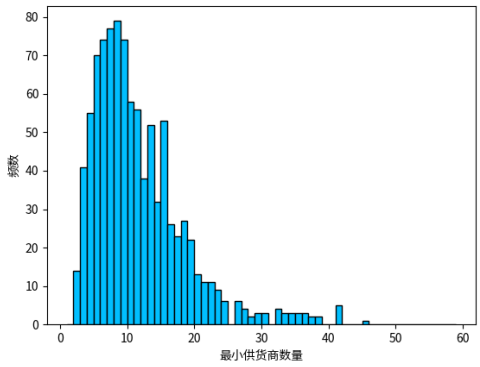


图 2　最小供货商数量的分布情况

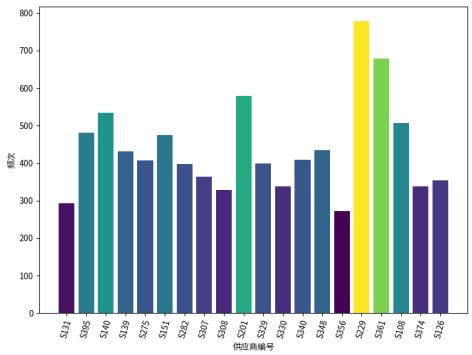


图 3　供货商的频次分布情况

根据上图求解最小供货商数量的均值，我们得到该企业应至少选择 11 家供应商供应原材料，具体的部分供货商的频次的分布情况如图3所示，我们可以选择频率最大的供应商作为实际进行供应的供应商。

根据上述频次分布情况得到的供应商组合为 S229、S361、S201、S140、S108、S395、S151、S348 、S139、S340、S275。这些选择的供应商都在问题一给出的前 50 家最重要

供应商的集合中。这说明我们的问题一的评价模型能较好的判断供应商对供货的影响情况。

接下来我们进行问题二中最小化总成本的优化模型的求解，我们使用 SAA 方法进行求解，令 SAA 方法中的采样数 $K = 1000$，对模型进行求解，一共得到 269 组可行解。其中最小订货价格和最小转运消耗产能的分布情况如下图所示：

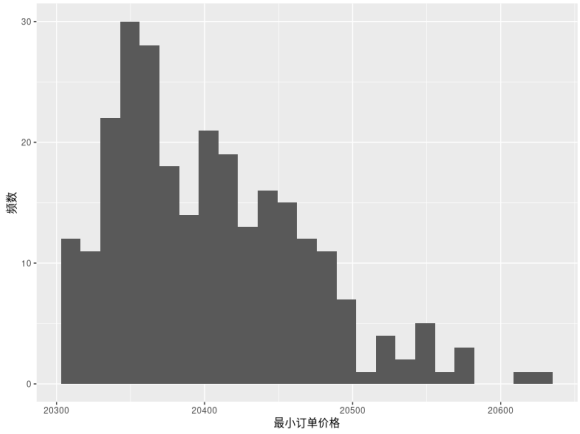


图 4　最小订货价格的分布情况

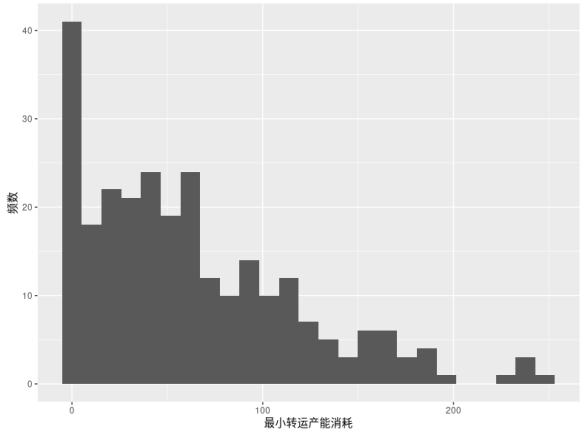


图 5　最小转运消耗产能的分布情况

最小消费价格的期望为 20403.2, 最小转运产能消耗的期望为 62.1。

针对上述的供应商组合，未来 24 周内厂商向 11 家供应商的总订货量以及各种类型原料的供货量占比如下图：

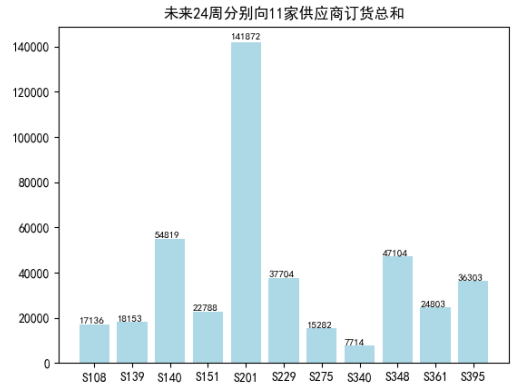


图 6　问题二中的具体订货量分配情况

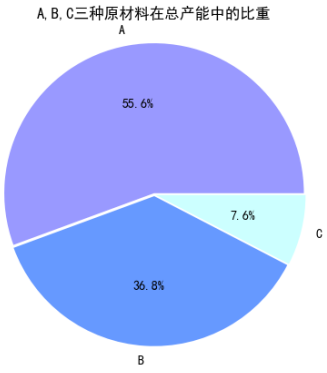


图 7　问题二中各类原料的订购占比

我们可以整理每周订货的原料对应的总价值和总产能，每周转运耗损的各种原料的总价值和总产能对比订货和转运过程中的产能情况来分析订购方案和转运方案的实施效果。

求解得到的每周订货的 A、B、C 类原材料的数量及每周转运耗损的 A、B、C 类原材料的数量的堆积图如下：

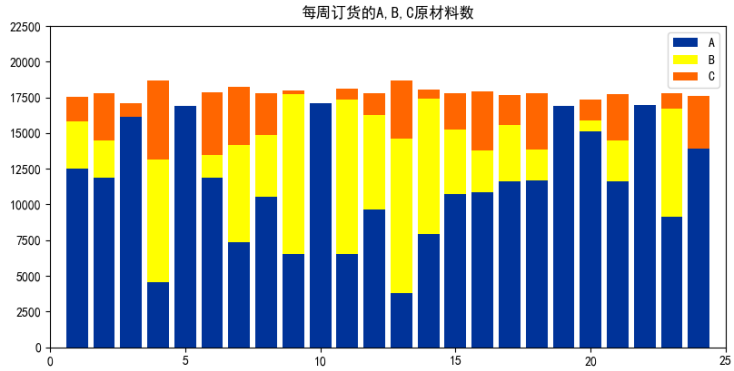


图 8　问题二中的每周订购情况

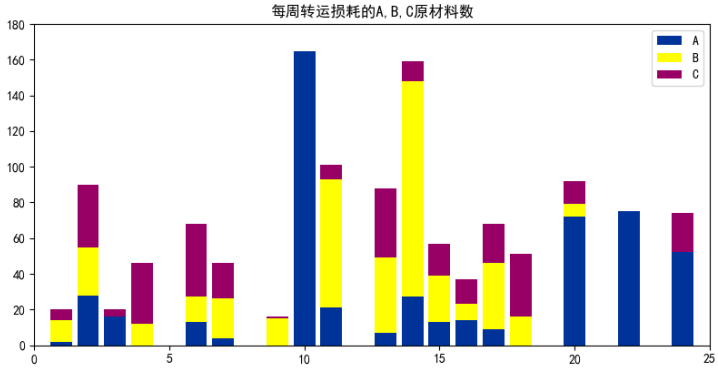


图 9　问题二每周转运损耗情况

24 周内的平均订单总价值为 20401，平均订单总产能为 28282；24 周内的平均耗损总价值为 59，平均耗损总产能为 82。平均耗损总价值占订单总价值的 0.2892%，这一损耗情况相对随机选取转运商时平均 1% 的损耗率明显更小，说明我们的优化模型针对最小化损耗率有较好的优化效果。

订单总产能减去耗损后对应的总产能为 28200，排除耗损影响后的产能恰好能满足企业的周产能需求，转运方案和订货方案的实施效果较好。

六、 问题三

6.1 模型建立

在问题三中，我们只需要对企业的生产成本和转运成本进行优化求解，不必考虑问题二中的供应商数量约束条件。此外，我们希望尽量多地采购 A 类和尽量少地采购 C 类原材料，从而让原料的体积最小化以减少转运及仓储的成本。

我们在问题三中继续使用样本平均近似法（SAA）生成 K 个具有代表性的场景，从而把随机规划问题转化为普通的线性规划问题。其中每种场景出现的可能性为 p_k，且 $\sum_{k\in K} p_k = 1$。我们定义两个指示变量 α 和 γ，$\alpha_i = 1$ 时，第 i 家供应商提供原料 A，否则 $\alpha = 0$。同理，$\gamma_i = 1$ 时，第 i 家供应商提供原料 C，否则 $\gamma = 0$。

用 I 表示全体供应商的集合，假设 x_{it} 表示第 t 周企业向供应商 i 的订货量，y_{ijt} 表示第 t 周转运商 j 运送供应商 i 的货量，w_{jt} 表示第 t 周转运商 j 的耗损率。

问题三中的优化问题的目标函数可表示为：

$$\min \sum_{k\in K}\sum_{i\in I} p_k x_{it}^k u_{b_i} \tag{25}$$

$$\max \sum_{k\in K}\sum_{i\in I} p_k \alpha_i x_{it}^k \tag{26}$$

$$\min \sum_{k\in K}\sum_{i\in I} p_k \gamma_i x_{it}^k \tag{27}$$

$$\min \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} p_k y_{ijt}^k w_{jt}^k u_{b_i} \tag{28}$$

公式25代表了让在各种情形下公司从供应商 i 处采购的采购成本的数学期望最小的目标函数，而公式26和公式27表示了尽可能多的采购 A，尽可能少的采购 C 的目标函数，而公式28代表使得运输损耗最小的目标函数。

由于问题三相对问题二去除了供应商数量的约束条件，因而供货商可以在所有供应商中进行选择。关于问题三的其它约束条件，考虑到我们必须满足企业的生产需求，所以一个约束条件可表示为在计算由于配送导致的损耗后，若各个供货商供货的的理论最大值能够满足每周产能时，供货对应的产能应大于等于企业每周的总产能，否则订货量应等于总供应量。即：

$$\begin{cases} \sum_{i\in I} \frac{x_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} \geq P \qquad \text{if} \quad \sum_{i\in I} \frac{G_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} > P \quad \forall t \in T; \forall k \in K \\ x_{it}^k = G_{it}^k \quad \text{else} \quad \forall i \in I; \forall t \in T; \forall k \in K \end{cases} \tag{29}$$

此外，供应商每周只能供应有限数量的货物，因而在理想情况下企业向供货商的订货量不应当超过供货商的总供应能力。这一约束可以表示为：

$$x_{it}^k \leq G_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \tag{30}$$

在关于原料运输过程中，供货商提供给转运商进行转运的来自某个订单的总原料量应当与企业的订货量相等，即：

$$\sum_{j\in J} y_{ijt}^k = x_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \tag{31}$$

此外，由于每家转运商的运输能力为 6000 立方米/周，我们应当限制每周内各家供应商的最大转运量

$$\sum_{i\in I} y_{ijt}^k \le 6000 \quad \forall j \in J; \forall t \in T; \forall k \in K \tag{32}$$

由于我们在选择供应商时增加了在集合 I_1 中选取的隐含条件，因而我们得到的订购方案和转运方案在是在使得供应商总数最少的供应商集合中选择的。

最小化总成本的随机规划模型可以整理如下：

$$\begin{aligned}
&\min \sum_{k\in K}\sum_{i\in I} p_k x_{it}^k u_{b_i} \\
&\max \sum_{k\in K}\sum_{i\in I} p_k \alpha_i x_{it}^k \\
&\min \sum_{k\in K}\sum_{i\in I} p_k \gamma_i x_{it}^k \\
&\min \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} p_k y_{ijt}^k w_{jt}^k u_{b_i} \\
\text{s.t.}\quad &\begin{cases} \sum_{i\in I} \frac{x_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} \ge P \qquad \text{if} \quad \sum_{i\in I} \frac{G_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} > P \quad \forall t \in T; \forall k \in K \\ x_{it}^k = G_{it}^k \quad \text{else} \quad \forall i \in I; \forall t \in T; \forall k \in K \end{cases} \\
&x_{it}^k \le G_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \\
&\sum_{j\in J} y_{ijt}^k = x_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \\
&\sum_{i\in I} y_{ijt}^k \le 6000 \quad \forall j \in J; \forall t \in T; \forall k \in K
\end{aligned}$$

6.2 模型求解

在问题三中，我们采取与问题二相似的分层序列法和 SAA 方法。在使用分层序列法处理多目标规划问题时，我们优先考虑体积最小的目标函数，之后考虑总订货价值最小的目标，最后考虑对模型整体影响最小的损耗最小的目标函数。之后我们用 SAA 方法用抽样的方法将随机变量用样本表示，结合两种方法最终把多目标随机规划问题转化为多个按顺序的线性规划问题进行求解。

我们通过 SAA 方法求解随机规划问题，令随机规划问题中的 $K = 1000$，对模型进行求解，一共得到了 967 组可行解。其中每周的 A 类和 C 类原材料的订货量的频数分布情况如下：

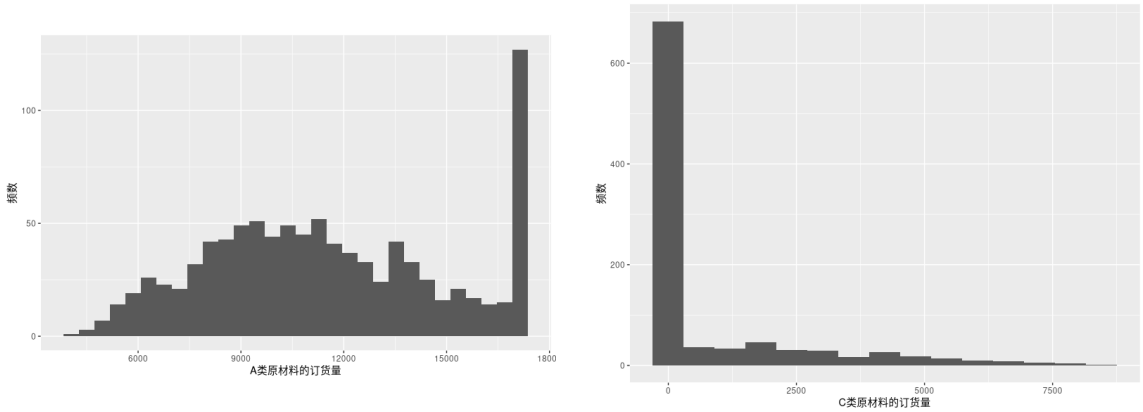


图 10　A 类和 C 类原材料订货量的频数分布直方图

求解得到 A 的订货量的数学期望为 11521,C 的订货量的数学期望为 885。

我们得到了一种可能的未来 24 周内每周订货和转运的分配方式，并得到了这种分配方式下每周对应的 A、B、C 原材料数及其总价值和总产能情况（详细数据见附录）。

其中 24 周内的平均订单总价值为 20389，24 周内平均订单总产能为 28263 立方米，A，B，C 在 24 周内的总订货量分别为 307837，105070，4367，其中 A、B、C 类订单的占比情况如图11。对比下面面的饼图11与饼图7可以发现问题三中求解结果内原材料 C 的比重明显下降。

问题三中的每周订货和损耗的 A、B、C 类原料数的堆积图如下：

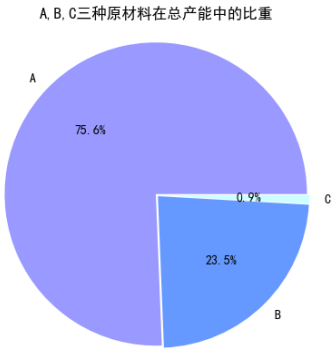


图 11　问题三中各类原料的订购占比

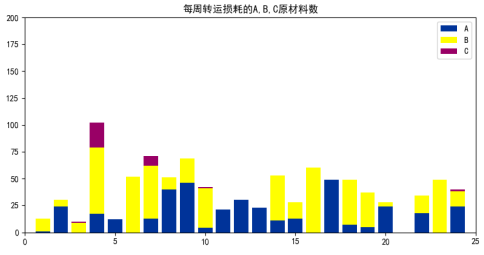


图 12　问题三中各类原料的损耗情况

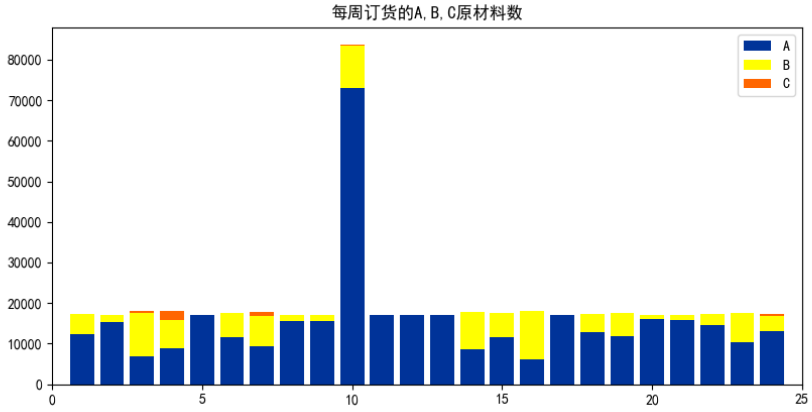


图 13　问题三中各类原料的订购情况

求解得到问题三的分配方案中中 24 周的平均耗损总价值为 45，平均耗损总产能为 62，平均耗损总价值占订单总价值的 0.219%。

订单总产能减去耗损总产能为 28201，能够满足企业的周产能需求，转运方案和订货方案的实施效果良好。

七、 问题四

7.1 模型建立

在问题四中，企业希望在据现有原材料的供应商和转运商的限制下尽可能的提高产能。在这种情况下，企业希望尽可能的收购所有供应商提供的货物来最大化企业的产能。一个目标函数可以表示为使企业的总产能最大，即：

$$\max \sum_{k\in K}\sum_{i\in I} p_k \frac{x_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} \tag{33}$$

由于我们使用服从正态分布的随机变量 Q 来表示供应商提供货物的供货量，我们的优化模型是一个随机规划模型。我们可以使用样本平均近似法（SAA）生成 K 个具有代表性的场景，从而在每个特定的场景内把随机规划问题转化为普通的线性规划问题。

由于对转运商的选择与不会影响企业对供应商的选择，且在转运过程中尽可能最小化转运损耗可以最大化企业实际的接收量，从而增加总产能，我们的目标函数也使转运过程中的转运总损耗最小，即：

$$\min \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} p_k y_{ijt}^k w_{jt}^k u_{b_i} \tag{34}$$

关于优化模型的约束条件，供应商每周只能供应有限数量的货物，因而在理想情况下企业向供货商的订货量不应当超过供货商的总供应能力，这一约束可以表示为：

$$x_{it}^k \le G_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \tag{35}$$

在原料运输过程中，供货商提供给转运商进行转运的来自某个订单的总原料量应当与企业的订货量相等，即：

$$\sum_{j\in J} y_{ijt}^k = x_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \tag{36}$$

此外，由于每家转运商的运输能力为 6000 立方米/周，我们应当限制每周内各家供应商的最大转运量：

$$\sum_{i\in I} y_{ijt}^k \leq 6000 \quad \forall j \in J; \forall t \in T; \forall k \in K \tag{37}$$

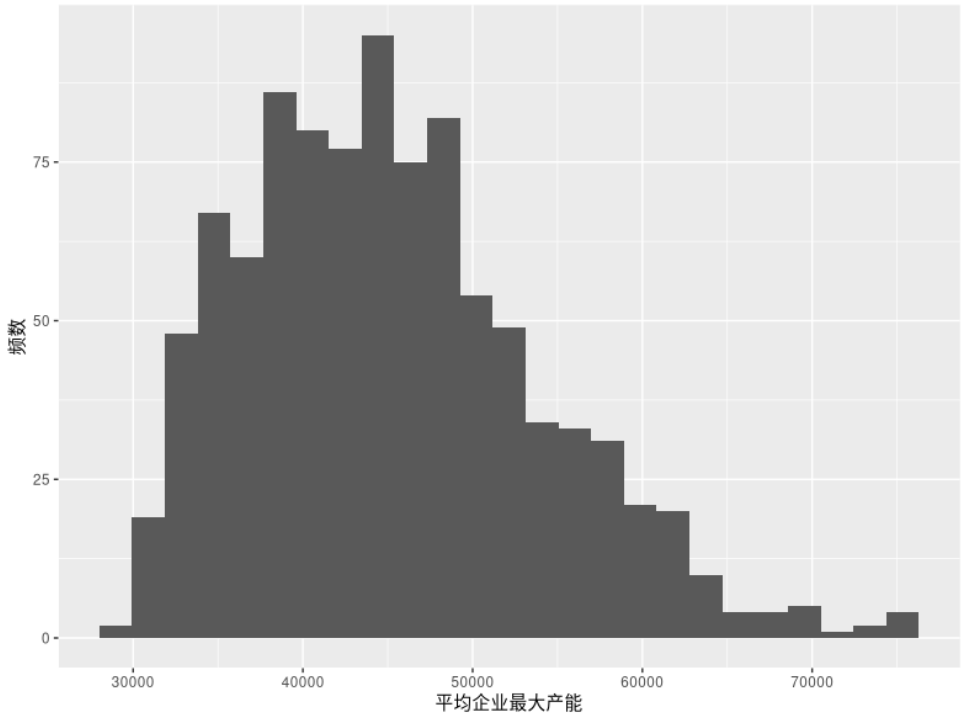
问题四中最大化生产能力的随机规划模型可以整理如下：

$$\begin{aligned}
&\max \sum_{k\in K}\sum_{i\in I} p_k \frac{x_{it}^k - \sum_{j\in J} y_{ijt}^k w_{jt}^k}{c_{b_i}} \\
&\min \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} p_k y_{ijt}^k w_{jt}^k u_{b_i} \\
\text{s.t.}\quad & x_{it}^k \leq G_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \\
& \sum_{j\in J} y_{ijt}^k = x_{it}^k \quad \forall i \in I; \forall t \in T; \forall k \in K \\
& \sum_{i\in I} y_{ijt}^k \leq 6000 \quad \forall j \in J; \forall t \in T; \forall k \in K
\end{aligned} \tag{38}$$

7.2 模型求解

在问题四中，我们继续采用与问题二相似的分层序列法和 SAA 方法求解多目标随机规划问题。我们优先考虑最大化总产能的目标函数，最后考虑对模型整体影响最小的损耗最小的目标函数。之后我们用 SAA 方法用抽样的方法将随机变量用样本表示，结合两种方法最终把多目标随机规划问题转化为多个线性规划问题进行求解。

在使用 SAA 方法的求解过程中，我们令 SAA 方法的采样数 $K = 1000$ 对模型进行求解，一共得到了 987 组可行解。各组可行解的频率分布情况如下图所示：



求解得到 24 周内的平均企业最大产能为 45159 立方米。在不考虑企业资金限制的情况下，最大可以考虑把企业产能提升到每周 45159 立方米。

我们求得了一种可能的未来 24 周内的订购方案和转运方案，具体分析如下：

问题四中的每周订货量以及损耗的 A、B、C 类原料数的堆积图如下：

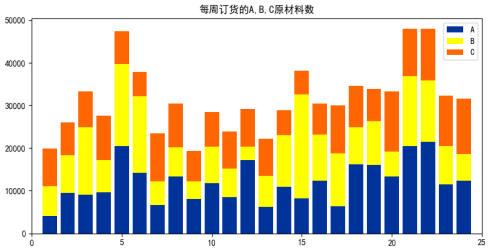


图 14　问题四各类原料的订购占比

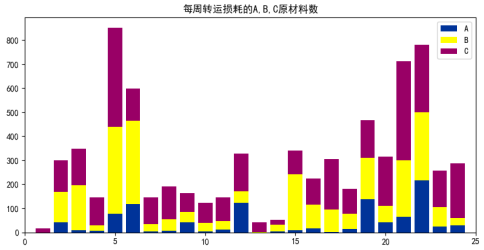


图 15　问题四中各类原料的损耗占比

在 24 周平均耗损总价值为：324，24 周内的平均耗损总产能为 449，平均耗损总价值占订单总价值的 0.925

可以看到，问题四中的损耗比例相比于问题二和问题三的转运的耗损比率进一步加大，这是因为随着企业总产能的最大化，原材料的需求增加，需要的转运量也增加，在转运的过程中，由于每家转运商转运量的限制，会被迫选择耗损率较大的转运商，从而导致平均耗损率增加。

八、 模型评价与推广

8.1 模型优点

1. 模型在问题二到四的优化模型中都建立了比较精确的规划模型，并给出了优化问题的最优解
2. 模型在问题二到问题四的求解中考虑到了供货上供货情况的随机波动，建立了随机规划模型，模型能够较好的考虑供货的随机因素

8.2 模型缺点

1. 在问题二中的 SW 检验方法只对比了 p 值的平均值，部分供货商的供货情况可能和样本整体的情况有所差异
2. 通过迭代求解随机规划模型的算法的时间复杂度较高

8.3 模型改进

根据文献 [4]，我们可以通过随机模拟方法处理机会约束条件, 并利用遗传算法的等启发式算法得到机会约束规划的目标函数最优值和决策变量最优解集。基于启发式算法的随机规划求解策略相对文章中采用的把随机规划转化为多个线性规划并进行求解的 SAA 方法的时间复杂度更低，更适用于对模型求解效率要求较高的场合。

参考文献

[1] 胡永宏. 对 TOPSIS 法用于综合评价的改进 [J]. 数学的实践与认识,2002(04):572-575.

[2] Deng Chunlin,Yang Liu. Sample Average Approximation Method for Chance Constrained Stochastic Programming in Transportation Model of Emergency Management[J]. Systems Engineering Procedia,2012,5:

[3] 李洪成. 数据的正态性检验方法及其统计软件实现 [J]. 统计与决策,2009(12):155-156.

[4] 陈高波, 刘海燕. 基于遗传算法的机会约束规划的区间估计 [J]. 西南交通大学学报,2004(05):687-690.

附录 A　问题一的求解结果

编号	类型	得分	编号	类型	得分
S229	A	0.018934	S031	B	0.004949
S361	C	0.015493	S365	C	0.004893
S140	B	0.015263	S374	C	0.004871
S108	B	0.012710	S364	B	0.004751
S282	A	0.010301	S178	A	0.004715
S151	C	0.009751	S169	B	0.004625
S275	A	0.009727	S221	A	0.004597
S329	A	0.009641	S040	B	0.004567
S340	B	0.009592	S174	B	0.004566
S139	B	0.008735	S266	A	0.004498
S131	B	0.008120	S342	C	0.004475
S308	B	0.007932	S367	B	0.004465
S330	B	0.007924	S218	C	0.004463
S268	C	0.007453	S037	C	0.004441
S356	C	0.007421	S338	B	0.004437
S306	C	0.007302	S175	B	0.004426
S352	A	0.006561	S237	A	0.004424
S194	C	0.006464	S346	B	0.004418
S348	A	0.006363	S324	B	0.004408
S143	A	0.006201	S080	C	0.004407
S395	A	0.005707	S294	C	0.004398
S307	A	0.005505	S141	B	0.004332
S247	C	0.005253	S005	A	0.004311
S284	C	0.005146	S007	A	0.004276
S201	A	0.005068	S092	B	0.004258

附录 B　问题二中的产能与订货情况

每周订货的 A、B、C 类原材料数及其对应的总价值和总产能为:

周数	1	2	3	4	5	6	7	8
A	12482	11899	16113	4561	16920	11851	7351	10555
B	3369	2577	0	8570	0	1587	6802	4318
C	1672	3312	991	5529	0	4423	4111	2928
总价格	20356	20426	20327	20429	20304	20390	20414	20344
总产能	28230	28336	28231	28266	28200	28299	28267	28201
周数	9	10	11	12	13	14	15	16
A	6496	17084	6502	9626	3763	7928	10750	10828
B	11228	1	10839	6662	10843	9455	4498	2945
C	277	0	789	1486	4053	651	2559	4138
总价格	20423	20502	20514	20365	20496	20565	20407	20371
总产能	28224	28475	28355	28201	28330	28443	28286	28256
周数	17	18	19	20	21	22	23	24
A	11603	16993	9135	13917	11611	11682	16920	15126
B	2890	0	7601	0	3949	2153	0	752
C	3228	2	1050	3688	2137	3989	0	1440
总价格	20331	20394	20373	20388	20414	20376	20304	20418
总产能	28200	28324	28200	28317	28303	28272	28200	28349

每周转运耗损的 A、B、C 类原料数及其总价值和总产能为:

周数	1	2	3	4	5	6	7	8	9	10	11	12
A	2	28	16	0	0	13	4	0	0	165	21	0
B	12	27	0	12	0	14	22	0	15	0	72	0
C	6	35	4	34	0	41	20	0	1	0	8	0
总价格	22	98	23	47	0	72	49	0	18	198	112	0
总产能	30	136	32	65	0	100	68	0	24	275	155	0
周数	13	14	15	16	17	18	19	20	21	22	23	24
A	7	27	13	14	9	0	0	72	0	75	0	52
B	42	121	26	9	37	16	0	7	0	0	0	0
C	39	11	18	14	22	35	0	13	0	0	0	22
总价格	94	177	62	41	74	53	0	107	0	90	0	84
总产能	129	244	86	56	102	73	0	149	0	125	0	117

附录 C　问题三中的产能与订货情况

每周订货的 A、B、C 类原材料数及其对应的总价值和总产能为:

周数	1	2	3	4	5	6	7	8
A	12256	15417	6789	8934	16933	11622	9339	15526
B	5143	1687	10689	6978	0	5881	7404	1590
C	0	0	507	2082	0	0	1099	0
总价格	20364	20356	20412	20479	20320	20416	20450	20380
总产能	28219	28251	28215	28354	28222	28281	28310	28286
周数	9	10	11	12	13	14	15	16
A	15430	7309	16942	16951	16943	8479	11503	6021
B	1713	10461	0	0	0	9339	5988	12050
C	0	168	0	0	0	0	0	0
总价格	20400	20446	20330	20341	20332	20448	20390	20480
总产能	28312	28265	28237	28252	28238	28282	28244	28293
周数	17	18	19	20	21	22	23	24
A	16970	12807	11842	16154	15831	14550	10271	13015
B	0	4575	5623	874	1198	2643	7363	3871
C	0	0	1	0	0	0	0	510
总价格	20364	20401	20397	20346	20315	20367	20424	20386
总产能	28283	28277	28258	28248	28200	28255	28274	28265

每周转运耗损的 A、B、C 类原料数及其总价值和总产能为:

周数	1	2	3	4	5	6	7	8	9	10	11	12
A	1	24	0	17	12	0	13	40	46	4	21	30
B	12	6	9	62	0	52	49	11	23	37	0	0
C	0	0	1	23	0	0	9	0	0	1	0	0
总价格	14	35	11	112	14	57	78	60	80	46	25	36
总产能	20	49	15	154	20	79	108	83	112	64	35	50
周数	13	14	15	16	17	18	19	20	21	22	23	24
A	23	11	13	0	49	7	5	24	0	18	0	24
B	0	42	15	60	0	42	32	4	0	16	49	14
C	0	0	0	0	0	0	0	0	0	0	0	2
总价格	28	59	32	66	59	55	41	33	0	39	54	46
总产能	38	82	44	91	82	75	57	46	0	54	74	64

附录 D　问题四中的产能与订货情况

每周转运耗损的 A、B、C 类原料数及其总价值和总产能为:

周数	1	2	3	4	5	6	7	8	9	10	11	12
A	0	41	10	7	77	118	3	6	42	4	11	122
B	0	128	186	23	362	346	30	49	43	36	36	49
C	16	130	153	115	413	135	113	135	78	84	98	157
总价格	16	320	370	149	904	657	150	196	176	128	151	357
总产能	22	443	511	206	1250	908	207	272	243	178	209	496
周数	13	14	15	16	17	18	19	20	21	22	23	24
A	0	4	10	16	1	15	138	41	64	216	23	30
B	2	28	231	98	95	63	173	68	237	283	82	29
C	40	20	99	109	208	103	156	206	411	283	152	228
总价格	42	56	365	236	314	190	512	330	748	854	270	296
总产能	59	77	504	327	434	264	709	457	1037	1182	374	411

每周订货的 A、B、C 类原材料数及其对应的总价值和总产能为:

周数	1	2	3	4	5	6	7	8
A	4024	9458	9112	9558	20484	14207	6591	13274
B	7085	8866	15788	7585	19235	17914	5573	6852
C	8824	7716	8420	10455	7646	5688	11312	10269
总价格	21446	28818	36721	30268	53385	42442	25352	33735
总产能	29697	39913	50802	41943	73903	58721	35140	46768
周数	9	10	11	12	13	14	15	16
A	8000	11828	8519	17241	6192	10849	8191	12295
B	4239	8523	6697	3085	7271	12188	24465	10926
C	7116	8110	8705	8846	8664	5798	5457	7258
总价格	21379	31679	26294	32929	24092	32224	42198	34031
总产能	29639	43891	36436	45695	33370	44601	58299	47127
周数	17	18	19	20	21	22	23	24
A	6284	16221	16033	13281	20469	21464	11445	12405
B	12487	8658	10273	5862	16333	14331	8997	6211
C	11207	9739	7623	14107	11198	12205	11862	13008
总价格	32484	38728	38163	36492	53727	53726	35493	34726
总产能	44958	53680	52874	50610	74415	74438	49182	48152

附录 E　用于数据预处理的 Python 代码

```
   #代码1：数据处理
import pandas as pd
import matplotlib.pyplot as plt

file_name_1 = '附件1 近5年402家供应商的相关数据.xls'

coefi_A = 0.6
coefi_B = 0.66
```

```
coefi_C = 0.72

data1 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=0)
data2 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=1)


#计算每周ABC三种原材料的订货量，以及他们对应的总产能
dinghuoABC = pd.DataFrame(index = ['A', 'B', 'C', '总产能'], columns = data1.columns)

for i in dinghuoABC.index:
    for c in dinghuoABC.columns:
        dinghuoABC.at[i, c] = 0
        for k in data1.index:
            if k[1] == i:
                dinghuoABC.at[i, c] += data1.at[k, c]

for c in dinghuoABC.columns:
    dinghuoABC.at['总产能', c] = dinghuoABC.at['A', c]/0.6 + dinghuoABC.at['B', c]/0.66 +
        dinghuoABC.at['C', c]/0.72


#计算每周ABC三种原材料的供货量，以及他们对应的总产能
gonghuoABC = pd.DataFrame(index = ['A', 'B', 'C', '总产能'], columns = data2.columns)

for i in gonghuoABC.index:
    for c in gonghuoABC.columns:
        gonghuoABC.at[i, c] = 0
        for k in data2.index:
            if k[1] == i:
                gonghuoABC.at[i, c] += data2.at[k, c]

for c in gonghuoABC.columns:
    gonghuoABC.at['总产能', c] = gonghuoABC.at['A', c]/0.6 + gonghuoABC.at['B', c]/0.66 +
        gonghuoABC.at['C', c]/0.72

#订货ABC
dinghuoABC = pd.DataFrame(index = ['A', 'B', 'C', '总产能'], columns = data1.columns)

for i in dinghuoABC.index:
    for c in dinghuoABC.columns:
        dinghuoABC.at[i, c] = 0
        for k in data1.index:
            if k[1] == i:
                dinghuoABC.at[i, c] += data1.at[k, c]

for c in dinghuoABC.columns:
    dinghuoABC.at['总产能', c] = dinghuoABC.at['A', c]/0.6 + dinghuoABC.at['B', c]/0.66 +
```

```
        dinghuoABC.at['C', c]/0.72

supplier_feature = pd.DataFrame(index = data1.index, columns = ['总订货数', '总供货数',
    '供/订', '订货周数', '供>=订周数'])

for i in supplier_feature.index:
    supplier_feature.at[i, '总订货数'] = 0
    for j in data1.columns:
        supplier_feature.at[i, '总订货数'] += data1.at[i, j]


for i in supplier_feature.index:
    supplier_feature.at[i, '总供货数'] = 0
    for j in data2.columns:
        supplier_feature.at[i, '总供货数'] += data2.at[i, j]

for i in supplier_feature.index:
    supplier_feature.at[i, '供/订'] = supplier_feature.at[i, '总供货数']/supplier_feature.at[i,
        '总订货数']

for i in supplier_feature.index:
    supplier_feature.at[i, '订货周数'] = 0
    for j in data1.columns:
        if data1.at[i, j] != 0:
            supplier_feature.at[i, '订货周数'] += 1

for i in supplier_feature.index:
    supplier_feature.at[i, '供>=订周数'] = 0
    for j in data1.columns:
        if data1.at[i, j] <= data2.at[i, j] and data1.at[i, j] != 0:
            supplier_feature.at[i, '供>=订周数'] += 1

#分为24周为一组，共10组
dinghuoABC24 = pd.DataFrame(index = ['A', 'B', 'C', '总产能'], columns = range(1, 11))
k = 0
for i in range(dinghuoABC24.shape[1]):
    for j in range(dinghuoABC24.shape[0]):
        dinghuoABC24.iloc[j, i] = dinghuoABC.iloc[:, k:k+24].sum(axis=1).iloc[j]
    k += 24

gonghuoABC24 = pd.DataFrame(index = ['A', 'B', 'C', '总产能'], columns = range(1, 11))
k = 0
for i in range(gonghuoABC24.shape[1]):
    for j in range(gonghuoABC24.shape[0]):
        gonghuoABC24.iloc[j, i] = gonghuoABC.iloc[:, k:k+24].sum(axis=1).iloc[j]
    k += 24
```

附录 F　问题一中用于数据预处理的 Python 代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob

def normalizing(array):
    max=array.max()
    min=array.min()
    return (array-min)/(max-min)

def standlizing(array):
    std=array.std()
    return array/std

orderf=pd.read_csv('order.csv')
providef=pd.read_csv('provide.csv')

ord=np.array(orderf)
prov=np.array(providef)
osum=[]
psum=[]
for i in range(ord.shape[0]):
    osu=np.sum(ord[i,2:])
    psu=np.sum(prov[i,2:])
    if ord[i][1]=='A':
        osum.append(osu/0.6)
    elif ord[i][1]=='B':
        osum.append(osu/0.66)
    else:
        osum.append(osu/0.72)
    if prov[i][1]=='A':
        psum.append(psu/0.6)
    elif prov[i][1]=='B':
        psum.append(psu/0.66)
    else:
        psum.append(psu/0.72)
osum=np.array(osum)
psum=np.array(psum)

oparr=[]

ord=ord[:,2:]
prov=prov[:,2:]
```

```
for i in range(ord.shape[0]):
    cntord = 0
    satiprov = 0
    for j in range(ord.shape[1]):
        if (ord[i][j]) > 0 and (ord[i][j] < prov[i][j]):
            satiprov += 1
        if ord[i][j] > 0:
            cntord += 1
    oparr.append(satiprov/cntord)

oparr=np.array(oparr)

dataf=pd.DataFrame({'tot_supply':psum,'supply/order(ordernum)':oparr,'supply/order(materialnum)':psum/osum})
dataf.to_csv('org_data_withmattcost.csv',index=False)
std_dataf=pd.DataFrame({'tot_supply':standlizing(psum),'supply/order(ordernum)':standlizing(oparr),'supply/orde
std_dataf.to_csv('std_data_withmattcost.csv',index=False)
proc_dataf=pd.DataFrame({'tot_supply':standlizing(psum)*0.7855600529132699,'supply/order(ordernum)':standlizing
proc_dataf.to_csv('proc_data_withmattcost.csv',index=False)
```

附录 G　问题一中用熵权法求权重的 Python 代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob

def normalizing(array):
    max=array.max()
    min=array.min()
    return (array-min)/(max-min)

dataf=pd.read_csv('std_data_withmattcost.csv')

tots=normalizing(np.array(dataf['tot_supply']));tots_sum=tots.sum()
soo=normalizing(np.array(dataf['supply/order(ordernum)']));soo_sum=soo.sum()
som=normalizing(np.array(dataf['supply/order(materialnum)']));som_sum=som.sum()

n=tots.shape[0]
k=1.0/np.log(n)

pmat=[]
for i in range(n):
    pmat.append([])
    if tots[i]==0:
        pmat[i].append(0)
```

```
    else:
        p=tots[i]/tots_sum
        pmat[i].append(-np.log(p)*p*k)
    if soo[i]==0:
        pmat[i].append(0)
    else:
        p=soo[i]/soo_sum
        pmat[i].append(-np.log(p)*p*k)
    if som[i]==0:
        pmat[i].append(0)
    else:
        p=som[i]/som_sum
        pmat[i].append(-np.log(p)*p*k)

pmat=np.array(pmat)
g=1-pmat.sum(axis=0)

wmat=[]
for i in range(3):
    wmat.append(g[i]/g.sum())

print(wmat)
```

附录 H　问题一中用 TOPSIS 法进行综合评价的 Python 代码

```
import numpy as np
import pandas as pd

def standa(data):
    std=np.sqrt(np.sum(pow(data,2),axis=1))
    for i in range(std.shape[0]):
        for j in range(data[i].shape[0]):
            data[i][j]/=std[i]
    return data

def norm(data):
    maxlist=np.max(data,axis=1)
    minlist=np.min(data,axis=1)
    maxdist=[]
    mindist=[]
    reslist=[]
    for i in range(data.shape[1]):
        maxsum=0
        minsum=0
        for j in range(data.shape[0]):
```

```
            maxsum+=np.power(data[j][i]-maxlist[j],2)
            minsum+=np.power(data[j][i]-minlist[j],2)
        maxdist.append(np.sqrt(maxsum))
        mindist.append(np.sqrt(minsum))
        reslist.append(mindist[i]/(mindist[i]+maxdist[i]))
    result=np.array(reslist)/np.sum(reslist)
    return result

dataf=pd.read_csv('proc_data_withmattcost.csv')
id=np.array(pd.read_csv('provide.csv')['id'])
type=np.array(pd.read_csv('provide.csv')['type'])
datarr=np.array([np.array(dataf['tot_supply']),np.array(dataf['supply/order(ordernum)']),np.array(dataf['supply
featurenum=datarr.shape[0]
tempres=datarr.copy()
tempres=standa(tempres)
tempres=norm(tempres)
res=pd.DataFrame({'id':id,'type':type,'score':tempres})
res=res.sort_values('score',ascending=False)
res.to_csv('rating_withmattcost.csv',index=False)
print(res)
```

附录 I　求解问题二中第一个随机规划模型的 Python 代码

```
    #问题2第1小题代码
import pandas as pd
import numpy as np
import random
import gurobipy
from gurobipy import quicksum as sum

file_name_1 = '附件1 近5年402家供应商的相关数据.xls'
file_name_2 = '附件2 近5年8家转运商的相关数据.xls'
data1 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=0)
data2 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=1)

data3 = pd.read_excel(file_name_2, index_col=[0], sheet_name=0)
samplenum = 200

############符号赋值
b = {}
idx = 1
for i in data1.index:
    b[idx] = i[1]
    idx += 1
```

```
coeffi = {'A': 0.6, 'B': 0.66, 'C': 0.72}
u = {'A': 1.2, 'B': 1.1, 'C': 1}

#每周产能
P = 28200
I = range(1, 403)
J = range(1, 9)  #转运公司的集合

#####################################################求供货量均值标准差
hat_g = []
for i in data1.index:
    hat_g_i = []
    for c in data1.columns:
        if data1.at[i, c] != 0:
            hat_g_i.append(data2.at[i, c])
    hat_g.append(hat_g_i)

E = []
Var = []
std = []
for i in range(len(hat_g)):
    E.append(np.mean(hat_g[i]))

for i in range(len(hat_g)):
    Var.append(np.var(hat_g[i]))

for i in range(len(hat_g)):
    std.append(np.std(hat_g[i]))
################################################

#####################################################求耗损率均值标准差
hat_w = []
for i in data3.index:
    hat_w_i = []
    for c in data3.columns:
        if data3.at[i, c] != 0:
            hat_w_i.append(data3.at[i, c] / 100)
    hat_w.append(hat_w_i)

E_w = []
Var_w = []
std_w = []
for i in range(len(hat_w)):
    E_w.append(np.mean(hat_w[i]))

for i in range(len(hat_w)):
    Var_w.append(np.var(hat_w[i]))
```

```
for i in range(len(hat_w)):
    std_w.append(np.std(hat_w[i]))
################################################
G = [[0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,round(np.random.normal(E[i-1], std[i-1]))) for i in I]
    for k in range(samplenum)]

w = [[0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,np.random.normal(E_w[i-1], std_w[i-1])) for i in J]
    for k in range(samplenum)]
#w = [[0,0,0,0,0,0,0,0,0]] + [[0]+[E_w[i-1] for i in J] for k in range(samplenum)]
I_1 = [0, 229, 361, 201, 140, 108, 395, 151, 348, 139, 340, 275]

#排除掉可能会无解的G和w
del_list = []
for k in range(1, samplenum+1):
    total = 0
    for i in I:
        total += G[k][i]/coeffi[b[i]]
    if total <= 1.05 * P:
        del_list.append(k)

#正序删除的话，删除后序号就变了
for k in reversed(del_list):
        del G[k]
        del w[k]
samplenum = len(G) - 1

MODEL = gurobipy.Model()

x = MODEL.addVars(range(1,samplenum+1), I, vtype=gurobipy.GRB.BINARY, name="x")
y = MODEL.addVars(range(1,samplenum+1), I, J, vtype=gurobipy.GRB.INTEGER, name='y')
#总供量是否满足产能
z = MODEL.addVars(range(1,samplenum+1), vtype=gurobipy.GRB.BINARY, name="z")

MODEL.update()
MODEL.setObjective(sum(1.0 / samplenum * x[k, i] for i in I for k in range(1,samplenum+1)),
    sense = gurobipy.GRB.MINIMIZE)
#if (sum((G[k1][i] - sum(y[k1, i, j] * w[k1][j] for j in J) )/ coeffi[b[i]] for i in I) > P
    for k1 in range(1,samplenum+1)):
#MODEL.addConstrs(z[k] = 0 for k in range(1,samplenum+1) if P >= sum((G[k][i] - sum(y[k, i, j]
    * w[k][j] for j in J) )/ coeffi[b[i]] for i in I) )
#MODEL.addConstrs( sum(x[k, i] for i in I ) >= 402 for k in range(1,samplenum+1) if P >=
    sum((G[k][i] - sum(y[k, i, j] * w[k][j] for j in J) )/ coeffi[b[i]] for i in I) )
MODEL.addConstrs(sum(( x[k,i] * G[k][i] - sum(y[k, i, j] * w[k][j] for j in J) )/ coeffi[b[i]]
    for i in I) >= P for k in range(1,samplenum+1))
MODEL.addConstrs(sum(y[k, i, j] for j in J) == x[k, i] * G[k][i] for i in I for k in
    range(1,samplenum+1))
```

```
MODEL.addConstrs(sum(y[k, i, j] for i in I) <= 6000 for j in J for k in range(1,samplenum+1))
MODEL.optimize()


#打印至少需要多少供应商
ynum_list = []
for k in range(1, samplenum+1):
    ynum = 0
    for i in I:
        if x[k, i].x:
            ynum += 1
    ynum_list.append(ynum)
print(ynum_list)
print(np.mean(ynum_list))
```

附录 J　求解问题二中第二个随机规划模型的 Python 代码

```
    #问题2第2小题代码
import pandas as pd
import numpy as np
import random
import gurobipy
from gurobipy import quicksum as sum

file_name_1 = '附件1 近5年402家供应商的相关数据.xls'
file_name_2 = '附件2 近5年8家转运商的相关数据.xls'
data1 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=0)
data2 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=1)

data3 = pd.read_excel(file_name_2, index_col=[0], sheet_name=0)
samplenum = 200

############符号赋值
b = {}
idx = 1
for i in data1.index:
    b[idx] = i[1]
    idx += 1

coeffi = {'A': 0.6, 'B': 0.66, 'C': 0.72}
u = {'A': 1.2, 'B': 1.1, 'C': 1}

#每周产能
P = 28200
I = range(1, 403)
```

```
J = range(1, 9)  #转运公司的集合

#####################################################求供货量均值标准差
hat_g = []
for i in data1.index:
    hat_g_i = []
    for c in data1.columns:
        if data1.at[i, c] != 0:
            hat_g_i.append(data2.at[i, c])
    hat_g.append(hat_g_i)

E = []
Var = []
std = []
for i in range(len(hat_g)):
    E.append(np.mean(hat_g[i]))

for i in range(len(hat_g)):
    Var.append(np.var(hat_g[i]))

for i in range(len(hat_g)):
    std.append(np.std(hat_g[i]))
################################################

#####################################################求耗损率均值标准差
hat_w = []
for i in data3.index:
    hat_w_i = []
    for c in data3.columns:
        if data3.at[i, c] != 0:
            hat_w_i.append(data3.at[i, c] / 100)
    hat_w.append(hat_w_i)

E_w = []
Var_w = []
std_w = []
for i in range(len(hat_w)):
    E_w.append(np.mean(hat_w[i]))

for i in range(len(hat_w)):
    Var_w.append(np.var(hat_w[i]))

for i in range(len(hat_w)):
    std_w.append(np.std(hat_w[i]))
################################################
```

```
G = [[0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,round(np.random.normal(E[i-1], std[i-1]))) for i in I]
    for k in range(samplenum)]

w = [[0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,np.random.normal(E_w[i-1], std_w[i-1])) for i in J]
    for k in range(samplenum)]
#w = [[0,0,0,0,0,0,0,0,0]] + [[0]+[E_w[i-1] for i in J] for k in range(samplenum)]
I_1 = [0, 229, 361, 201, 140, 108, 395, 151, 348, 139, 340, 275]

########################################排除掉可能会无解的G和w
del_list = []
for k in range(1, samplenum+1):
   total = 0
   for i in I_1:
      if i > 0:
         total += G[k][i]/coeffi[b[i]]
   if total <= 1.05 * P:
      del_list.append(k)

#正序删除的话，删除后序号就变了
for k in reversed(del_list):
      del G[k]
      del w[k]
samplenum = len(G) - 1
################################################



MODEL = gurobipy.Model()

x = MODEL.addVars(range(1,samplenum+1), range(1, len(I_1)), vtype=gurobipy.GRB.INTEGER,
    name="x")
y = MODEL.addVars(range(1,samplenum+1), range(1, len(I_1)), J, vtype=gurobipy.GRB.INTEGER,
    name='y')

MODEL.update()

# 创建目标函数
MODEL.setObjectiveN(sum(1.0 / samplenum * x[k, i] * u[b[I_1[i]]] for i in range(1, len(I_1))
    for k in range(1,samplenum+1)), priority=1, index=0)
MODEL.setObjectiveN(sum(1.0 / samplenum * y[k, i, j] * w[k][j] * u[b[I_1[i]]] for i in
    range(1, len(I_1)) for j in J for k in range(1,samplenum+1)), priority=0, index=1)

MODEL.addConstrs(sum(( x[k,i] - sum(y[k, i, j] * w[k][j] for j in J) )/ coeffi[b[I_1[i]]] for
    i in range(1, len(I_1))) >= P for k in range(1,samplenum+1))
MODEL.addConstrs(x[k, i] <= G[k][I_1[i]] for i in range(1, len(I_1)) for k in
    range(1,samplenum+1))
MODEL.addConstrs(sum(y[k, i, j] for j in J) == x[k, i] for i in range(1, len(I_1)) for k in
```

```
    range(1,samplenum+1))
MODEL.addConstrs(sum(y[k, i, j] for i in range(1, len(I_1))) <= 6000 for j in J for k in
    range(1,samplenum+1))

MODEL.optimize()


# #打印至少需要多少供应商
# ynum_list = []
# for k in range(1, samplenum+1):
#     ynum = 0
#     for i in I:
#         if x[k, i].x:
#             ynum += 1
#     ynum_list.append(ynum)
# print(ynum_list)

# for i in range(1, 12):
#     sum = 0
#     for j in range(1, 25):
#         sum += x[j, i].x
#     print(sum)

# Adlist = []
# Bdlist = []
# Cdlist = []
# for i in range(1, 25):
#     A = 0
#     B = 0
#     C = 0
#     for j in range(1, 12):
#         if b[I_1[j]] == 'A':
#             A += x[i, j].x
#         elif b[I_1[j]] == 'B':
#             B += x[i, j].x
#         elif b[I_1[j]] == 'C':
#             C += x[i, j].x
#     Adlist.append(A)
#     Bdlist.append(B)
#     Cdlist.append(C)

# dinghuo_analysis = pd.DataFrame(index = ['A', 'B', 'C', '总价格', '总产能'], columns =
    range(1, 25))
# for j in range(dinghuo_analysis.shape[1]):
#     dinghuo_analysis.iloc[0, j] = Adlist[j]
#     dinghuo_analysis.iloc[1, j] = Bdlist[j]
#     dinghuo_analysis.iloc[2, j] = Cdlist[j]
```

```
#    dinghuo_analysis.iloc[3, j] = round(1.2 * dinghuo_analysis.iloc[0, j] + 1.1 *
    dinghuo_analysis.iloc[1, j] + 1 * dinghuo_analysis.iloc[2, j])
#    dinghuo_analysis.iloc[4, j] = round(dinghuo_analysis.iloc[0, j] / 0.6 +
    dinghuo_analysis.iloc[1, j] / 0.66 + dinghuo_analysis.iloc[2, j] / 0.72)

# Awlist = []
# Bwlist = []
# Cwlist = []
# for i in range(1, 25):
#    A = 0
#    B = 0
#    C = 0
#    for j in range(1, 12):
#        for k in range(1, 9):
#            if b[I_1[j]] == 'A':
#                A += y[i, j, k].x * w[i][k]
#            elif b[I_1[j]] == 'B':
#                B += y[i, j, k].x * w[i][k]
#            elif b[I_1[j]] == 'C':
#                C += y[i, j, k].x * w[i][k]
#    Awlist.append(round(A))
#    Bwlist.append(round(B))
#    Cwlist.append(round(C))

# waste_analysis = pd.DataFrame(index = ['A', 'B', 'C', '总价格', '总产能'], columns = range(1,
    25))
# for j in range(waste_analysis.shape[1]):
#    waste_analysis.iloc[0, j] = Awlist[j]
#    waste_analysis.iloc[1, j] = Bwlist[j]
#    waste_analysis.iloc[2, j] = Cwlist[j]
#    waste_analysis.iloc[3, j] = round(1.2 * waste_analysis.iloc[0, j] + 1.1 *
    waste_analysis.iloc[1, j] + 1 * waste_analysis.iloc[2, j])
#    waste_analysis.iloc[4, j] = round(waste_analysis.iloc[0, j] / 0.6 +
    waste_analysis.iloc[1, j] / 0.66 + waste_analysis.iloc[2, j] / 0.72)
```

附录 K　问题二中检验各个供应商的产能分布情况的 R 语言代码

```
library(tidyverse)
library(openxlsx)
library(fitdistrplus)
data<- read.xlsx("附件1 近5年402家供应商的相关数据.xlsx",sheet=2)
#data0<- read.xlsx("附件2 近5年8家转运商的相关数据.xlsx")
#data0<- data[-1]
data<-data[c(-1,-2)]
#datat<- data-data0/data
```

```
res<-list()
for (i in seq_len(length(unlist(data[,1])))) {
    d<-unlist(data[i,])
    d[d==0]<-NA
    d<-na.omit(d)
     if (length(d)<=5) {
         break
     }
    sa<-sample(d,5)
    #d<-d[order(d,decreasing=TRUE)[1:length(d)]]

    #d<-order(d,decreasing = TRUE)[1:j]
    #print(d)
    #descdist(d)
    #res[i]<-ks.test(d,rlnorm(100,mean(d),var(d)))$p.value
    res[i]<-shapiro.test(unlist(sa))$p.value
    #print(shapiro.test(d))
}
print(mean(unlist(res)))
```

附录 L　在问题 2 中用于数据可视化的 Python 代码

```
import matplotlib.pyplot as plt
import numpy as np

week=[i for i in range(1,25)]
provider=['S108', 'S139', 'S140', 'S151', 'S201', 'S229', 'S275', 'S340', 'S348', 'S361',
    'S395']

tot_provide=[17136, 18153, 54819, 22788, 141872, 37704, 15282, 7714, 47104, 24803, 36303]

amount_A=[12482, 11899, 16113, 4561, 16920, 11851, 7351, 10555, 6496, 17084,
        6502, 9626, 3763, 7928, 10750, 10828, 11611, 11682, 16920, 15126,
        11603, 16993, 9135, 13917]
amount_B=[3369, 2577, 0, 8570, 0, 1587, 6802, 4318, 11228, 1, 10839, 6662,
        10843, 9455, 4498, 2945, 3949, 2153, 0, 752, 2890, 0, 7601, 0]
amount_C=[1672, 3312, 991, 5529, 0, 4423, 4111, 2928, 277, 0, 789, 1486,
        4053, 651, 2559, 4138, 2137, 3989, 0, 1440, 3228, 2, 1050, 3688]
tot_price=[20356, 20426, 20327, 20429, 20304, 20390, 20414, 20344, 20423,
         20502, 20514, 20365, 20496, 20565, 20407, 20371, 20414, 20376,
         20304, 20418, 20331, 20394, 20373, 20388]
tot_produce=[28230, 28336, 28231, 28266, 28200, 28299, 28267, 28201, 28224,
           28475, 28355, 28201, 28330, 28443, 28286, 28256, 28303, 28272,
           28200, 28349, 28200, 28324, 28200, 28317]
```

```
worn_A=[2, 28, 16, 0, 0, 13, 4, 0, 0, 165, 21, 0, 7, 27, 13, 14, 9, 0, 0, 72, 0, 75, 0, 52]
worn_B=[12, 27, 0, 12, 0, 14, 22, 0, 15, 0, 72, 0, 42, 121, 26, 9, 37, 16, 0, 7, 0, 0, 0, 0]
worn_C=[6, 35, 4, 34, 0, 41, 20, 0, 1, 0, 8, 0, 39, 11, 18, 14, 22, 35, 0, 13, 0, 0, 0, 22]
tot_worn_price=[22, 98, 23, 47, 0, 72, 49, 0, 18, 198, 112, 0, 94, 177, 62, 41, 74, 53, 0,
    107, 0, 90, 0, 84]
tot_worn_produce=[30, 136, 32, 65, 0, 100, 68, 0, 24, 275, 155, 0, 129, 244, 86, 56, 102, 73,
    0, 149, 0, 125, 0, 117]

plt.bar(provider,tot_provide,color='lightblue')
xlist=[i for i in range(1,12)]
for i in range(len(xlist)):
    xy=(xlist[i]-1.4,tot_provide[i]*1.01)
    text=str(tot_provide[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('未来24周分别向11家供应商订货总和')
plt.show()
plt.close()

plt.bar(week,amount_A,color='#003399',label='A')
plt.bar(week,amount_B,bottom=amount_A,color='#FFFF00',label='B')
plt.bar(week,amount_C,bottom=np.array(amount_A)+np.array(amount_B),color='#FF6600',label='C')
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料数')
plt.xlim([0,25])
plt.ylim([0,22500])
plt.legend()
plt.show()
plt.close()

plt.bar(week,tot_price,color='tomato')
for i in range(len(week)):
    xy=(week[i]-0.3,tot_price[i]*1.01)
    text=str(tot_price[i])
    plt.annotate(text=text,xy=xy,fontsize=5)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料对应的总价格')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,tot_produce,color='lightgreen')
for i in range(len(week)):
    xy=(week[i]-0.3,tot_produce[i]*1.01)
    text=str(tot_produce[i])
    plt.annotate(text=text,xy=xy,fontsize=5)
plt.rcParams['font.sans-serif']='SimHei'
```

```
plt.title('每周订货的A,B,C原材料对应的总产能')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,worn_A,color='#003399',label='A')
plt.bar(week,worn_B,bottom=worn_A,color='#FFFF00',label='B')
plt.bar(week,worn_C,bottom=np.array(worn_A)+np.array(worn_B),color='#990066',label='C')
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料数')
plt.xlim([0,25])
plt.ylim([0,180])
plt.legend()
plt.show()
plt.close()

plt.bar(week,tot_worn_price,color='sandybrown')
for i in range(len(week)):
    xy=(week[i]-0.1,tot_worn_price[i]*1.01)
    text=str(tot_worn_price[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料对应的总价格')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,tot_worn_produce,color='darkviolet')
for i in range(len(week)):
    xy=(week[i]-0.2,tot_worn_produce[i]*1.01)
    text=str(tot_worn_produce[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料对应的总产能')
plt.xlim([0,25])
plt.show()
plt.close()
```

附录 M　在问题 2-4 中用于数据可视化的 R 语言代码

```
library(ggplot2)

#data<-read.csv('3-A.txt',sep=",")
data<-read.csv('3-C.txt',sep=",")
names(data)<- c("list","num")
```

```r
p<-ggplot(data = data,aes(x=num))
#频次分布直方图
p+geom_histogram(bins=15)+labs(x="A类原材料的订货量",y="频数")
```

附录 N　在问题 3 中求解随机规划模型的 Python 代码

```python
#问题3求解代码
import pandas as pd
import numpy as np
import random
import gurobipy
from gurobipy import quicksum as sum

file_name_1 = '附件1 近5年402家供应商的相关数据.xls'
file_name_2 = '附件2 近5年8家转运商的相关数据.xls'
data1 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=0)
data2 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=1)

data3 = pd.read_excel(file_name_2, index_col=[0], sheet_name=0)
samplenum = 50

############符号赋值
b = {}
idx = 1
for i in data1.index:
    b[idx] = i[1]
    idx += 1

coeffi = {'A': 0.6, 'B': 0.66, 'C': 0.72}
u = {'A': 1.2, 'B': 1.1, 'C': 1}

#每周产能
P = 28200
I = range(1, 403)
J = range(1, 9)  #转运公司的集合

#########################################################求供货量均值标准差
hat_g = []
for i in data1.index:
    hat_g_i = []
    for c in data1.columns:
        if data1.at[i, c] != 0:
            hat_g_i.append(data2.at[i, c])
    hat_g.append(hat_g_i)
```

```
E = []
Var = []
std = []
for i in range(len(hat_g)):
    E.append(np.mean(hat_g[i]))

for i in range(len(hat_g)):
    Var.append(np.var(hat_g[i]))

for i in range(len(hat_g)):
    std.append(np.std(hat_g[i]))
################################################

#####################################################求耗损率均值标准差
hat_w = []
for i in data3.index:
    hat_w_i = []
    for c in data3.columns:
        if data3.at[i, c] != 0:
            hat_w_i.append(data3.at[i, c] / 100)
    hat_w.append(hat_w_i)

E_w = []
Var_w = []
std_w = []
for i in range(len(hat_w)):
    E_w.append(np.mean(hat_w[i]))

for i in range(len(hat_w)):
    Var_w.append(np.var(hat_w[i]))

for i in range(len(hat_w)):
    std_w.append(np.std(hat_w[i]))
################################################


G = [[0,0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,round(np.random.normal(E[i-1], std[i-1]))) for i in I]
     for k in range(samplenum)]

w = [[0,0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,np.random.normal(E_w[i-1], std_w[i-1])) for i in J]
     for k in range(samplenum)]
#w = [[0,0,0,0,0,0,0,0,0,0]] + [[0]+[E_w[i-1] for i in J] for k in range(samplenum)]
#I_1 = [0, 229, 361, 201, 140, 108, 395, 151, 348, 139, 340, 275]
I_1 =[0] + [i for i in I]
#########################################排除掉可能会无解的G和w
del_list = []
for k in range(1, samplenum+1):
```

```
    total = 0
    for i in I_1:
        if i > 0:
            total += G[k][i]/coeffi[b[i]]
    if total <= 1.05 * P:
        del_list.append(k)

#正序删除的话，删除后序号就变了
for k in reversed(del_list):
        del G[k]
        del w[k]
samplenum = len(G) - 1
##################################################

##################指示变量#########################
aaa = [0]
bbb = [0]
ccc = [0]
for i in I:
    if b[i] == 'A':
        aaa.append(1)
        bbb.append(0)
        ccc.append(0)
    elif b[i] == 'B':
        aaa.append(0)
        bbb.append(1)
        ccc.append(0)
    elif b[i] == 'C':
        aaa.append(0)
        bbb.append(0)
        ccc.append(1)
##########################################


MODEL = gurobipy.Model()

x = MODEL.addVars(range(1,samplenum+1), range(1, len(I_1)), vtype=gurobipy.GRB.INTEGER,
    name="x")
y = MODEL.addVars(range(1,samplenum+1), range(1, len(I_1)), J, vtype=gurobipy.GRB.INTEGER,
    name='y')

MODEL.update()

# 创建目标函数
MODEL.setObjectiveN(sum(1.0 / samplenum * x[k, i] * u[b[I_1[i]]] for k in range(1,samplenum+1)
    for i in I), index=0)
#MODEL.setObjectiveN(sum(1.0 / samplenum * x[k, i] for k in range(1,samplenum+1) for i in I),
```

```
    index=0)
MODEL.setObjectiveN(-sum(1.0 / samplenum * aaa[i] * x[k, i] for k in range(1,samplenum+1) for
    i in I), index=1)
MODEL.setObjectiveN(sum(1.0 / samplenum * ccc[i] * x[k, i] for k in range(1,samplenum+1) for i
    in I), index=2)
MODEL.setObjectiveN(sum(1.0 / samplenum * y[k, i, j] * w[k][j] * u[b[I_1[i]]] for k in
    range(1,samplenum+1) for i in I for j in J), index=3)

MODEL.addConstrs(sum(( x[k,i] - sum(y[k, i, j] * w[k][j] for j in J) )/ coeffi[b[I_1[i]]] for
    i in range(1, len(I_1))) >= P for k in range(1,samplenum+1))
MODEL.addConstrs(x[k, i] <= G[k][I_1[i]] for i in range(1, len(I_1)) for k in
    range(1,samplenum+1))
MODEL.addConstrs(sum(y[k, i, j] for j in J) == x[k, i] for i in range(1, len(I_1)) for k in
    range(1,samplenum+1))
MODEL.addConstrs(sum(y[k, i, j] for i in range(1, len(I_1))) <= 6000 for j in J for k in
    range(1,samplenum+1))

MODEL.optimize()


# #打印至少需要多少供应商
# ynum_list = []
# for k in range(1, samplenum+1):
#     ynum = 0
#     for i in I:
#         if x[k, i].x:
#             ynum += 1
#     ynum_list.append(ynum)
# print(ynum_list)

# for i in range(1, 12):
#     sum = 0
#     for j in range(1, 25):
#         sum += x[j, i].x
#     print(sum)

# Adlist = []
# Bdlist = []
# Cdlist = []
# for i in range(1, 25):
#     A = 0
#     B = 0
#     C = 0
#     for j in range(1, 403):
#         if b[I_1[j]] == 'A':
#             A += x[i, j].x
#         elif b[I_1[j]] == 'B':
```

```
#           B += x[i, j].x
#       elif b[I_1[j]] == 'C':
#           C += x[i, j].x
#    Adlist.append(round(A))
#    Bdlist.append(round(B))
#    Cdlist.append(round(C))

# dinghuo_analysis = pd.DataFrame(index = ['A', 'B', 'C', '总价格', '总产能'], columns =
    range(1, 25))
# for j in range(dinghuo_analysis.shape[1]):
#    dinghuo_analysis.iloc[0, j] = Adlist[j]
#    dinghuo_analysis.iloc[1, j] = Bdlist[j]
#    dinghuo_analysis.iloc[2, j] = Cdlist[j]
#    dinghuo_analysis.iloc[3, j] = round(1.2 * dinghuo_analysis.iloc[0, j] + 1.1 *
    dinghuo_analysis.iloc[1, j] + 1 * dinghuo_analysis.iloc[2, j])
#    dinghuo_analysis.iloc[4, j] = round(dinghuo_analysis.iloc[0, j] / 0.6 +
    dinghuo_analysis.iloc[1, j] / 0.66 + dinghuo_analysis.iloc[2, j] / 0.72)

# Awlist = []
# Bwlist = []
# Cwlist = []
# for i in range(1, 25):
#    A = 0
#    B = 0
#    C = 0
#    for j in range(1, 403):
#        for k in range(1, 9):
#            if b[I_1[j]] == 'A':
#                A += y[i, j, k].x * w[i][k]
#            elif b[I_1[j]] == 'B':
#                B += y[i, j, k].x * w[i][k]
#            elif b[I_1[j]] == 'C':
#                C += y[i, j, k].x * w[i][k]
#    Awlist.append(round(A))
#    Bwlist.append(round(B))
#    Cwlist.append(round(C))

# waste_analysis = pd.DataFrame(index = ['A', 'B', 'C', '总价格', '总产能'], columns = range(1,
    25))
# for j in range(waste_analysis.shape[1]):
#    waste_analysis.iloc[0, j] = Awlist[j]
#    waste_analysis.iloc[1, j] = Bwlist[j]
#    waste_analysis.iloc[2, j] = Cwlist[j]
#    waste_analysis.iloc[3, j] = round(1.2 * waste_analysis.iloc[0, j] + 1.1 *
    waste_analysis.iloc[1, j] + 1 * waste_analysis.iloc[2, j])
#    waste_analysis.iloc[4, j] = round(waste_analysis.iloc[0, j] / 0.6 +
    waste_analysis.iloc[1, j] / 0.66 + waste_analysis.iloc[2, j] / 0.72)
```

附录 O　在问题 3 用于数据可视化的 Python 代码

```
import matplotlib.pyplot as plt
import numpy as np

week=[i for i in range(1,25)]

amount_A=[12256, 15417, 6789, 8934, 16933, 11622, 9339, 15526, 15430, 73099,
        16942, 16951, 16943, 8479, 11503, 6021, 16970, 12807, 11842, 16154,
        15831, 14550, 10271, 13015]
amount_B=[5143, 1687, 10689, 6978, 0, 5881, 7404, 1590, 1713, 10461,
        0, 0, 0, 9339, 5988, 12050, 0, 4575, 5623, 874,
        1198, 2643, 7363, 3871]
amount_C=[0, 0, 507, 2082, 0, 0, 1099, 0, 0, 168,
        0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
        0, 0, 0, 510]
tot_price=[20364, 20356, 20412, 20479, 20320, 20416, 20450, 20380, 20400, 20446,
         20330, 20341, 20332, 20448, 20390, 20480, 20364, 20401, 20397, 20346,
         20315, 20367, 20424, 20386]
tot_produce=[28219, 28251, 28215, 28354, 28222, 28281, 28310, 28286, 28312, 28265,
           28237, 28252, 28238, 28282, 28244, 28293, 28283, 28277, 28258, 28248,
           28200, 28255, 28274, 28265]

worn_A=[1, 24, 0, 17, 12, 0, 13, 40, 46, 4, 21, 30, 23, 11, 13, 0, 49, 7, 5, 24, 0, 18, 0, 24]
worn_B=[12, 6, 9, 62, 0, 52, 49, 11, 23, 37, 0, 0, 0, 42, 15, 60, 0, 42, 32, 4, 0, 16, 49, 14]
worn_C=[0, 0, 1, 23, 0, 0, 9, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]
tot_worn_price=[14, 35, 11, 112, 14, 57, 78, 60, 80, 46, 25, 36, 28, 59, 32, 66, 59, 55, 41,
    33, 0, 39, 54, 46]
tot_worn_produce=[20, 49, 15, 154, 20, 79, 108, 83, 112, 64, 35, 50, 38, 82, 44, 91, 82, 75,
    57, 46, 0, 54, 74, 64]

labels=['A', 'B', 'C']
weight_prob2=[55.6, 36.8, 7.6]
weight_prob3=[0.7563, 0.2347, 0.009]

plt.pie(weight_prob2,explode=[0.01,0.01,0.01],labels=labels,autopct='%1.1f%%',colors=['#9999FF','#6699FF','#CCF
plt.rcParams['font.sans-serif']='SimHei'
plt.title('A,B,C三种原材料在总产能中的比重')
plt.axis('equal')
plt.show()
plt.close()

plt.pie(weight_prob3,explode=[0.01,0.01,0.01],labels=labels,autopct='%1.1f%%',colors=['#9999FF','#6699FF','#CCF
```

```
plt.rcParams['font.sans-serif']='SimHei'
plt.title('A,B,C三种原材料在总产能中的比重')
plt.axis('equal')
plt.show()
plt.close()

plt.bar(week,amount_A,color='#003399',label='A')
plt.bar(week,amount_B,bottom=amount_A,color='#FFFF00',label='B')
plt.bar(week,amount_C,bottom=np.array(amount_A)+np.array(amount_B),color='#FF6600',label='C')
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料数')
plt.xlim([0,25])
plt.legend()
plt.show()
plt.close()

plt.bar(week,tot_price,color='tomato')
for i in range(len(week)):
    xy=(week[i]-0.3,tot_price[i]*1.01)
    text=str(tot_price[i])
    plt.annotate(text=text,xy=xy,fontsize=5)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料对应的总价格')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,tot_produce,color='lightgreen')
for i in range(len(week)):
    xy=(week[i]-0.3,tot_produce[i]*1.01)
    text=str(tot_produce[i])
    plt.annotate(text=text,xy=xy,fontsize=5)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料对应的总产能')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,worn_A,color='#003399',label='A')
plt.bar(week,worn_B,bottom=worn_A,color='#FFFF00',label='B')
plt.bar(week,worn_C,bottom=np.array(worn_A)+np.array(worn_B),color='#990066',label='C')
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料数')
plt.xlim([0,25])
plt.ylim([0,200])
plt.legend()
plt.show()
```

```
plt.close()

plt.bar(week,tot_worn_price,color='sandybrown')
for i in range(len(week)):
    xy=(week[i]-0.1,tot_worn_price[i]*1.01)
    text=str(tot_worn_price[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料对应的总价格')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,tot_worn_produce,color='darkviolet')
for i in range(len(week)):
    xy=(week[i]-0.2,tot_worn_produce[i]*1.01)
    text=str(tot_worn_produce[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料对应的总产能')
plt.xlim([0,25])
plt.show()
plt.close()
```

附录 P　在问题 4 中用于求解随机规划模型的 Python 代码

```
#问题4模型求解
import pandas as pd
import numpy as np
import random
import gurobipy
from gurobipy import quicksum as sum

file_name_1 = '附件1 近5年402家供应商的相关数据.xls'
file_name_2 = '附件2 近5年8家转运商的相关数据.xls'
data1 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=0)
data2 = pd.read_excel(file_name_1, index_col=[0, 1], sheet_name=1)

data3 = pd.read_excel(file_name_2, index_col=[0], sheet_name=0)
samplenum = 50

############符号赋值
b = {}
idx = 1
for i in data1.index:
```

```
    b[idx] = i[1]
    idx += 1

coeffi = {'A': 0.6, 'B': 0.66, 'C': 0.72}
u = {'A': 1.2, 'B': 1.1, 'C': 1}

#每周产能
P = 28200
I = range(1, 403)
J = range(1, 9)  #转运公司的集合

#####################################################求供货量均值标准差
hat_g = []
for i in data1.index:
    hat_g_i = []
    for c in data1.columns:
        if data1.at[i, c] != 0:
            hat_g_i.append(data2.at[i, c])
    hat_g.append(hat_g_i)

E = []
Var = []
std = []
for i in range(len(hat_g)):
    E.append(np.mean(hat_g[i]))

for i in range(len(hat_g)):
    Var.append(np.var(hat_g[i]))

for i in range(len(hat_g)):
    std.append(np.std(hat_g[i]))
################################################

#####################################################求耗损率均值标准差
hat_w = []
for i in data3.index:
    hat_w_i = []
    for c in data3.columns:
        if data3.at[i, c] != 0:
            hat_w_i.append(data3.at[i, c] / 100)
    hat_w.append(hat_w_i)

E_w = []
Var_w = []
std_w = []
for i in range(len(hat_w)):
    E_w.append(np.mean(hat_w[i]))
```

```
for i in range(len(hat_w)):
    Var_w.append(np.var(hat_w[i]))

for i in range(len(hat_w)):
    std_w.append(np.std(hat_w[i]))
################################################


G = [[0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,round(np.random.normal(E[i-1], std[i-1]))) for i in I]
     for k in range(samplenum)]

w = [[0,0,0,0,0,0,0,0,0]] + [[0]+[max(0,np.random.normal(E_w[i-1], std_w[i-1])) for i in J]
     for k in range(samplenum)]
#w = [[0,0,0,0,0,0,0,0,0]] + [[0]+[E_w[i-1] for i in J] for k in range(samplenum)]
#I_1 = [0, 229, 361, 201, 140, 108, 395, 151, 348, 139, 340, 275]
I_1 =[0] + [i for i in I]
########################################排除掉可能会无解的G和w
del_list = []
for k in range(1, samplenum+1):
    total = 0
    for i in I_1:
        if i > 0:
            total += G[k][i]/coeffi[b[i]]
    if total <= 1.05 * P:
        del_list.append(k)

#正序删除的话，删除后序号就变了
for k in reversed(del_list):
        del G[k]
        del w[k]
samplenum = len(G) - 1
################################################




MODEL = gurobipy.Model()

x = MODEL.addVars(range(1,samplenum+1), range(1, len(I_1)), vtype=gurobipy.GRB.INTEGER,
     name="x")
y = MODEL.addVars(range(1,samplenum+1), range(1, len(I_1)), J, vtype=gurobipy.GRB.INTEGER,
     name='y')

MODEL.update()

# 创建目标函数
```

```
# MODEL.setObjectiveN(sum(1.0 / samplenum * x[k, i] * u[b[I_1[i]]] for k in
    range(1,samplenum+1) for i in I), index=1)
# MODEL.setObjectiveN(sum(1.0 / samplenum * x[k, i] for k in range(1,samplenum+1) for i in I),
    index=0)
# #MODEL.setObjectiveN(sum(1.0 / samplenum * y[k, i, j] * w[k][j] * u[b[I_1[i]]] for k in
    range(1,samplenum+1) for i in I for j in J), index=2)
#
# MODEL.addConstrs(sum(( x[k,i] - sum(y[k, i, j] * w[k][j] for j in J) )/ coeffi[b[I_1[i]]]
    for i in range(1, len(I_1))) >= P for k in range(1,samplenum+1))
# MODEL.addConstrs(x[k, i] <= G[k][I_1[i]] for i in range(1, len(I_1)) for k in
    range(1,samplenum+1))
# MODEL.addConstrs(sum(y[k, i, j] for j in J) == x[k, i] for i in range(1, len(I_1)) for k in
    range(1,samplenum+1))
# MODEL.addConstrs(sum(y[k, i, j] for i in range(1, len(I_1))) <= 6000 for j in J for k in
    range(1,samplenum+1))
MODEL.setObjectiveN(-sum((x[k,i] - sum(y[k, i, j] * w[k][j] for j in J)) / coeffi[b[I_1[i]]]
    for i in I for k in range(1,samplenum+1)), priority=1, index=0)
MODEL.setObjectiveN(sum(1.0 / samplenum * y[k, i, j] * w[k][j] * u[b[I_1[i]]] for k in
    range(1,samplenum+1) for i in I for j in J), priority=0, index=1)

MODEL.addConstrs(x[k, i] <= G[k][I_1[i]] for k in range(1,samplenum+1) for i in I)
MODEL.addConstrs(sum(y[k, i, j] for j in J) == x[k, i] for k in range(1,samplenum+1) for i in
    I)
MODEL.addConstrs(sum(y[k, i, j] for i in I) <= 6000 for j in J for k in range(1,samplenum+1))

MODEL.optimize()



# #打印至少需要多少供应商
# ynum_list = []
# for k in range(1, samplenum+1):
#     ynum = 0
#     for i in I:
#         if x[k, i].x:
#             ynum += 1
#     ynum_list.append(ynum)
# print(ynum_list)

# for i in range(1, 12):
#     sum = 0
#     for j in range(1, 25):
#         sum += x[j, i].x
#     print(sum)

# Adlist = []
# Bdlist = []
```

```
# Cdlist = []
# for i in range(1, 25):
#     A = 0
#     B = 0
#     C = 0
#     for j in range(1, 403):
#         if b[I_1[j]] == 'A':
#             A += x[i, j].x
#         elif b[I_1[j]] == 'B':
#             B += x[i, j].x
#         elif b[I_1[j]] == 'C':
#             C += x[i, j].x
#     Adlist.append(round(A))
#     Bdlist.append(round(B))
#     Cdlist.append(round(C))

# dinghuo_analysis = pd.DataFrame(index = ['A', 'B', 'C', '总价格', '总产能'], columns =
    range(1, 25))
# for j in range(dinghuo_analysis.shape[1]):
#     dinghuo_analysis.iloc[0, j] = Adlist[j]
#     dinghuo_analysis.iloc[1, j] = Bdlist[j]
#     dinghuo_analysis.iloc[2, j] = Cdlist[j]
#     dinghuo_analysis.iloc[3, j] = round(1.2 * dinghuo_analysis.iloc[0, j] + 1.1 *
    dinghuo_analysis.iloc[1, j] + 1 * dinghuo_analysis.iloc[2, j])
#     dinghuo_analysis.iloc[4, j] = round(dinghuo_analysis.iloc[0, j] / 0.6 +
    dinghuo_analysis.iloc[1, j] / 0.66 + dinghuo_analysis.iloc[2, j] / 0.72)

# Awlist = []
# Bwlist = []
# Cwlist = []
# for i in range(1, 25):
#     A = 0
#     B = 0
#     C = 0
#     for j in range(1, 403):
#         for k in range(1, 9):
#             if b[I_1[j]] == 'A':
#                 A += y[i, j, k].x * w[i][k]
#             elif b[I_1[j]] == 'B':
#                 B += y[i, j, k].x * w[i][k]
#             elif b[I_1[j]] == 'C':
#                 C += y[i, j, k].x * w[i][k]
#     Awlist.append(round(A))
#     Bwlist.append(round(B))
#     Cwlist.append(round(C))

# waste_analysis = pd.DataFrame(index = ['A', 'B', 'C', '总价格', '总产能'], columns = range(1,
```

```
    25))
# for j in range(waste_analysis.shape[1]):
#     waste_analysis.iloc[0, j] = Awlist[j]
#     waste_analysis.iloc[1, j] = Bwlist[j]
#     waste_analysis.iloc[2, j] = Cwlist[j]
#     waste_analysis.iloc[3, j] = round(1.2 * waste_analysis.iloc[0, j] + 1.1 *
    waste_analysis.iloc[1, j] + 1 * waste_analysis.iloc[2, j])
#     waste_analysis.iloc[4, j] = round(waste_analysis.iloc[0, j] / 0.6 +
    waste_analysis.iloc[1, j] / 0.66 + waste_analysis.iloc[2, j] / 0.72)
```

附录 Q　在问题 4 中用于数据可视化的 Python 代码

```
import matplotlib.pyplot as plt
import numpy as np

week=[i for i in range(1,25)]

amount_A=[4024, 9458, 9112, 9558, 20484, 14207, 6591, 13274, 8000, 11828,
        8519, 17241, 6192, 10849, 8191, 12295, 6284, 16221, 16033, 13281,
        20469, 21464, 11445, 12405]
amount_B=[7085, 8866, 15788, 7585, 19235, 17914, 5573, 6852, 4239, 8523,
        6697, 3085, 7271, 12188, 24465, 10926, 12487, 8658, 10273, 5862,
        16333, 14331, 8997, 6211]
amount_C=[8824, 7716, 8420, 10455, 7646, 5688, 11312, 10269, 7116, 8110,
        8705, 8846, 8664, 5798, 5457, 7258, 11207, 9739, 7623, 14107,
        11198, 12205, 11862, 13008]
tot_price=[21446, 28818, 36721, 30268, 53385, 42442, 25352, 33735, 21379, 31679,
         26294, 32929, 24092, 32224, 42198, 34031, 32484, 38728, 38163, 36492 ,
         53727, 53726, 35493, 34726]
tot_produce=[29697, 39913, 50802, 41943, 73903, 58721, 35140, 46768, 29639, 43891,
           36436, 45695, 33370, 44601, 58299, 47127, 44958, 53680, 52874, 50610,
           74415, 74438, 49182, 48152]

worn_A=[0, 41, 10, 7, 77, 118, 3, 6, 42, 4, 11, 122, 0, 4, 10, 16, 1, 15, 138, 41, 64, 216,
    23, 30]
worn_B=[0, 128, 186, 23, 362, 346, 30, 49, 43, 36, 36, 49, 2, 28, 231, 98, 95, 63, 173, 68,
    237, 283, 82, 29]
worn_C=[16, 130, 153, 115, 413, 135, 113, 135, 78, 84, 98, 157, 40, 20, 99, 109, 208, 103,
    156, 206, 411, 283, 152, 228]
tot_worn_price=[16, 320, 370, 149, 904, 657, 150, 196, 176, 128, 151, 357, 42, 56, 365, 236,
    314, 190, 512, 330, 748, 854, 270, 296]
tot_worn_produce=[22, 443, 511, 206, 1250, 908, 207, 272, 243, 178, 209, 496, 59, 77, 504,
    327, 434, 264, 709, 457, 1037, 1182, 374, 411]

plt.bar(week,amount_A,color='#003399',label='A')
```

```
plt.bar(week,amount_B,bottom=amount_A,color='#FFFF00',label='B')
plt.bar(week,amount_C,bottom=np.array(amount_A)+np.array(amount_B),color='#FF6600',label='C')
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料数')
plt.xlim([0,25])
plt.legend()
plt.show()
plt.close()

plt.bar(week,tot_price,color='tomato')
for i in range(len(week)):
    xy=(week[i]-0.3,tot_price[i]*1.01)
    text=str(tot_price[i])
    plt.annotate(text=text,xy=xy,fontsize=5)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料对应的总价格')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,tot_produce,color='lightgreen')
for i in range(len(week)):
    xy=(week[i]-0.3,tot_produce[i]*1.01)
    text=str(tot_produce[i])
    plt.annotate(text=text,xy=xy,fontsize=5)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周订货的A,B,C原材料对应的总产能')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,worn_A,color='#003399',label='A')
plt.bar(week,worn_B,bottom=worn_A,color='#FFFF00',label='B')
plt.bar(week,worn_C,bottom=np.array(worn_A)+np.array(worn_B),color='#990066',label='C')
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料数')
plt.xlim([0,25])
plt.legend()
plt.show()
plt.close()

plt.bar(week,tot_worn_price,color='sandybrown')
for i in range(len(week)):
    xy=(week[i]-0.1,tot_worn_price[i]*1.01)
    text=str(tot_worn_price[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
```

```
plt.title('每周转运损耗的A,B,C原材料对应的总价格')
plt.xlim([0,25])
plt.show()
plt.close()

plt.bar(week,tot_worn_produce,color='darkviolet')
for i in range(len(week)):
    xy=(week[i]-0.2,tot_worn_produce[i]*1.01)
    text=str(tot_worn_produce[i])
    plt.annotate(text=text,xy=xy,fontsize=8)
plt.rcParams['font.sans-serif']='SimHei'
plt.title('每周转运损耗的A,B,C原材料对应的总产能')
plt.xlim([0,25])
plt.show()
plt.close()
```