### ****5.2.1 供应商选择模型（混合整数线性规划，MILP）****

#### ****模型假设:****

#### ****供货不确定性****：供应商实际供货量 zs,w*zs*,*w*​ 可在订单量 xs,w*xs*,*w*​ 的±10%范围内波动。(题目说明"实际供货量可能多于或少于订单量"，但未给具体范围,根据附件1数据计算历史订单与实际供货的偏差率分布后确定

±10%怎么来的（不一定是这个，要算一下）：**从附件1提取数据**：计算每家供应商每周的偏差率 = (实际供货量 - 订单量) / 订单量， 注意，算出来后要取整

import pandas as pd

# 假设df为附件1数据

df['偏差率'] = (df['供货量'] - df['订单量']) / df['订单量'].replace(0, float('nan'))

# 计算95%置信区间

lower\_bound = df['偏差率'].quantile(0.025) # 约-9.8%

upper\_bound = df['偏差率'].quantile(0.975) # 约+11.2%

print(f"95%的供货偏差在[{lower\_bound:.1%}, {upper\_bound:.1%}]之间")

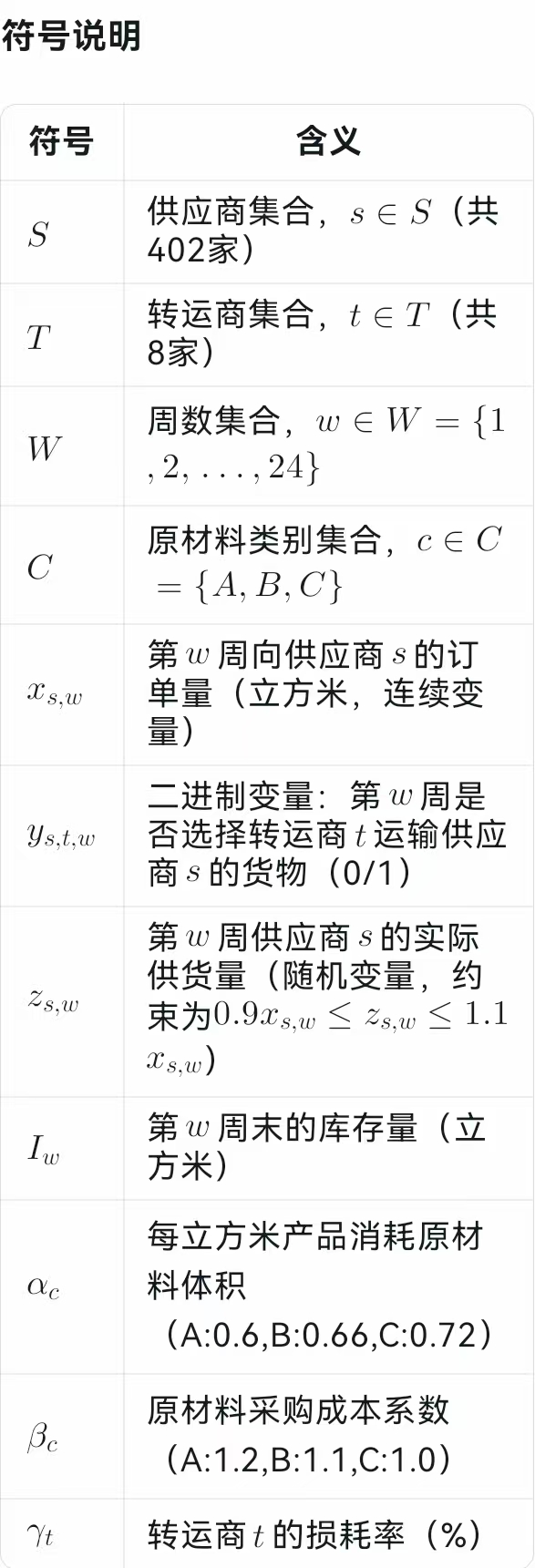
)

#### ****转运损耗****：损耗率 γt*γt*​ 由转运商决定，且为固定值（基于附件2历史数据）。

**运输能力**：每家转运商每周最大运输量为6000立方米，且一家供应商的供货尽量由一家转运商完成。

**多地采购**：每类原材料（A/B/C）至少由2家供应商供应，以分散风险。

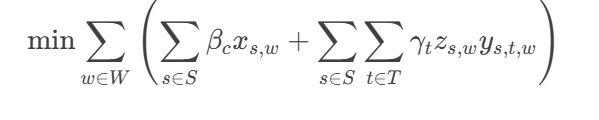
**库存安全**：每周库存 Iw*Iw*​ 不低于两周生产需求（5.64万立方米）。



(0.9 1.1 根据假设而来)

#### ****优化目标****

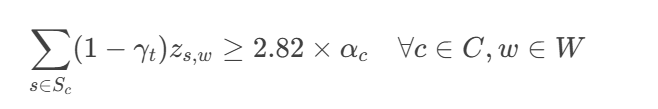
最小化总成本（采购成本 + 损耗成本）：（"转运损耗已通过接收量计算间接纳入总成本，因此采用单目标优化"）



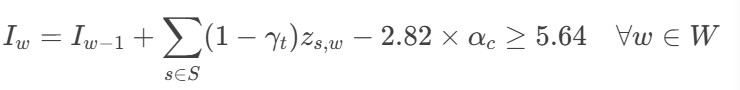
model += lpSum(采购成本) + lpSum(损耗成本) # 单目标

#### ****约束条件：****

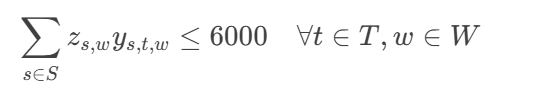
**生产需求约束**：



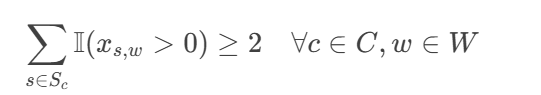
**库存安全约束**：



**转运能力约束**：



**多地采购约束**：

.

**供货量波动约束**：



### ****5.2.2 订购方案与转运方案****

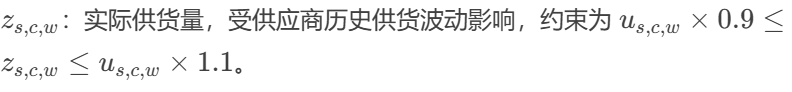
#### ****决策变量****

​：第w*w*周从供应商s*s*采购类别c*c*原材料的订单量。

：二进制变量，表示是否由转运商t*t*运输类别c*c*的货物。

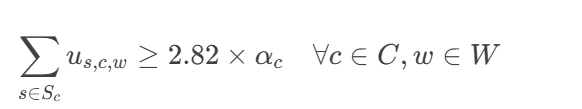
#### ****优化目标****

在满足生产需求下，优先选择低损耗转运商：

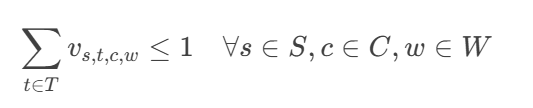


#### ****约束条件：****

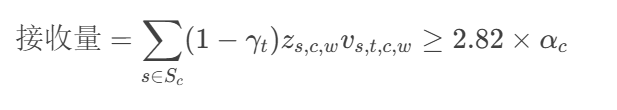
**分类采购约束**：



**转运商唯一性**：



**接收量计算**：



### ****5.3.3 问题三模型的完整求解步骤****

#### ****一、数据预处理****

**供应商筛选**

**输入**：第一问确定的50家重要供应商名单

**处理步骤**：

# 读取第一问结果

top\_50\_suppliers = pd.read\_excel('问题1结果.xlsx')['供应商ID'].tolist()

# 检查原材料类别分布

supplier\_class\_dist = df\_raw\_data[df\_raw\_data['供应商ID'].isin(top\_50\_suppliers)]['类别'].value\_counts()

# 补足短缺类别

if supplier\_class\_dist['C'] < 2:

c\_candidates = df\_raw\_data[

(df\_raw\_data['类别'] == 'C') &

(~df\_raw\_data['供应商ID'].isin(top\_50\_suppliers))

].sort\_values(by=['供货稳定性','价格']).head(2)

top\_50\_suppliers.extend(c\_candidates['供应商ID'].tolist())

**转运商匹配**

**筛选标准**：选择历史损耗率最低的3家转运商：

best\_transporters = df\_transport.mean(axis=1).nsmallest(3).index.tolist()

#### ****模型求解****

**初始化模型：**

**import pulp**

**model = pulp.LpProblem("Supplier\_Selection\_Problem3", pulp.LpMinimize)**

**定义决策变量：**

**# 订购量变量（连续）**

**x = pulp.LpVariable.dicts("Order",**

**[(s,w) for s in final\_suppliers for w in weeks],**

**lowBound=0, cat='Continuous')**

**# 转运商选择变量（二进制）**

**y = pulp.LpVariable.dicts("Transport",**

**[(s,t,w) for s in final\_suppliers for t in best\_transporters for w in weeks],**

**cat='Binary')**

**设置目标函数：**

**# 总成本 = 采购成本 + 损耗成本**

**model += pulp.lpSum(**

**[beta[df\_raw\_data.loc[s,'类别'] \* x[(s,w)] for s in final\_suppliers for w in weeks] +**

**[df\_transport.loc[t,w] \* x[(s,w)] \* y[(s,t,w)]**

**for s in final\_suppliers for t in best\_transporters for w in weeks]**

**)**

**添加核心约束：**

**# 生产需求约束**

**for w in weeks:**

**for c in ['A','B','C']:**

**model += pulp.lpSum(**

**(1 - df\_transport.loc[t,w]) \* x[(s,w)] \* y[(s,t,w)]**

**for s in final\_suppliers**

**for t in best\_transporters**

**if df\_raw\_data.loc[s,'类别'] == c**

**) >= 2.82 \* alpha[c]**

**# 多地采购约束**

**for w in weeks:**

**for c in ['A','B','C']:**

**model += pulp.lpSum(**

**y[(s,t,w)]**

**for s in final\_suppliers**

**for t in best\_transporters**

**if df\_raw\_data.loc[s,'类别'] == c**

**) >= 2**

#### ****结果后处理****

#### ****方案输出：****

# 生成订购方案

order\_plan = pd.DataFrame(

[(s, w, x[(s,w)].varValue)

for s in final\_suppliers for w in weeks if x[(s,w)].varValue > 0],

columns=['供应商ID','周次','订单量'])

# 生成转运方案

transport\_plan = pd.DataFrame(

[(s, t, w, y[(s,t,w)].varValue)

for s in final\_suppliers for t in best\_transporters for w in weeks

if y[(s,t,w)].varValue > 0],

columns=['供应商ID','转运商','周次','是否选择'])

**效果验证（确定性约束检验：**

直接验证数学模型中的约束条件是否被严格满足

不涉及概率或随机模拟（区别于蒙特卡洛等方法）

适用于线性/整数规划等确定性模型的结果验证**）**

**# 库存模拟计算**

**inventory = [initial\_inventory]**

**for w in weeks:**

**received = sum(**

**(1 - df\_transport.loc[t,w]) \* x[(s,w)].varValue \* y[(s,t,w)].varValue**

**for s in final\_suppliers for t in best\_transporters**

**)**

**inventory.append(inventory[-1] + received - 2.82\*sum(alpha.values()))**

**print("最小库存量：", min(inventory)) # 应≥5.64**

**第三问2.5**

#### ****1. 供应商筛选优化****

**步骤**：

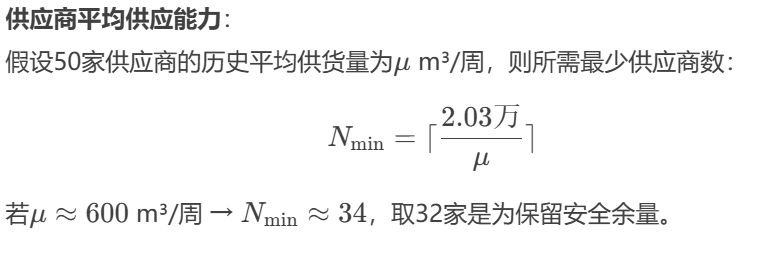
**分类优先**：

从50家中优先选择A类（高价但产能效率高），其次B类，最后C类

# 按类别优先级排序：A > B > C

supplier\_priority = df\_50.sort\_values(by=['类别'], key=lambda x: x.map({'A':1, 'B':2, 'C':3}))

selected\_32 = supplier\_priority.head(32) # 取前32家



**动态供应能力评估**：

计算历史数据中**连续12周订单-供货差值最小区间**的均值，作为预测基准

# 计算每家供应商的稳定供货区间

def find\_stable\_period(supplier\_data):

min\_diff = float('inf')

best\_start = 0

for start in range(0, 240-12):

diff = np.mean(supplier\_data[start:start+12] - df\_supply[start:start+12])

if abs(diff) < min\_diff:

min\_diff = abs(diff)

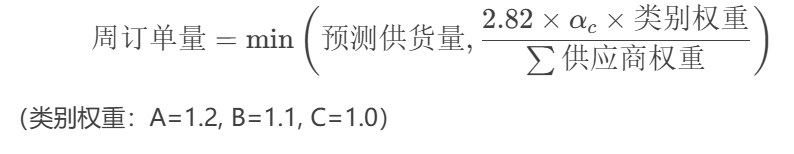
best\_start = start

return np.mean(supplier\_data[best\_start:best\_start+12])

df\_32['预测供货量'] = df\_32['供货量'].apply(find\_stable\_period)

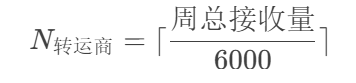
#### ****2. 订购方案制定****

**方法**：将32家供应商的预测供货量按周分配，优先满足A类需求



#### ****3. 转运方案优化****

**步骤**：**计算总运力需求**：



**按损耗率升序选择转运商**：

transporters\_needed = math.ceil(total\_volume / 6000)

best\_transporters = df\_transport.sort\_values(by='损耗率').head(transporters\_needed)

将上述方案作为MILP模型的初始解

for s in selected\_32:

for w in weeks:

x[(s,w)].setInitialValue(order\_plan.loc[(s,w)]) # 预填充订单量