5.2.1 供应商选择模型（ILP）

模型假设：

1. 供应商历史供货能力稳定

2. 转运损耗率使用附件2的平均值

3. 企业必须全量收购供应商供货

4. 同一供应商的原材料尽量由一家转运商运输

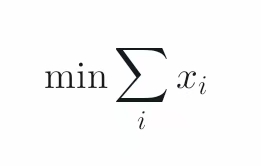
符号说明：



决策变量：

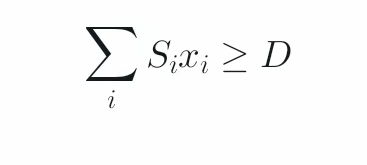


优化目标：

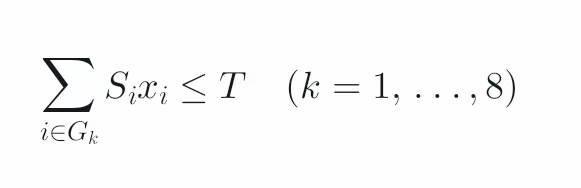


约束条件：

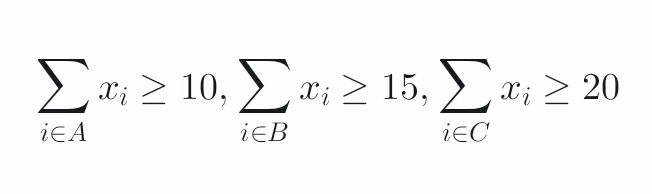
1. 需求：



2. 转运：



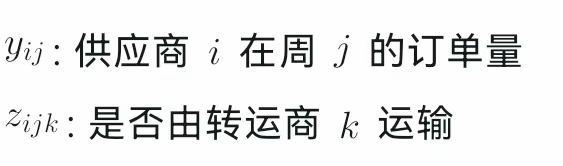
3. 类型平衡（在选择的供应商中，确保A、B、C三类原材料供应商的数量满足最低比例要求。这是为了防止过度依赖某一类原材料导致供应链风险。）：



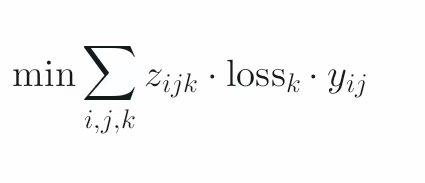
（比例怎么来的：按之前算的分别为30%，33%，36%，乘完50后大概分别为15，17，18，换成10，15，20是为了保证供应安全，略高于理论值）

5.2.2 订购与转运方案

决策变量：



优化目标：



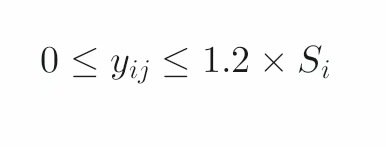
（最小化因转运损耗导致的原材料损失总量）

随机变量：



约束条件：

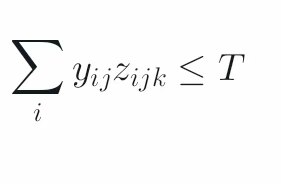
1. 订单限制：



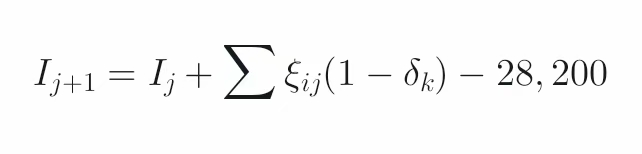
（允许20%超额订购）

（安全系数1.2是供应链管理的常用参数（可参考文献《供应链库存管理与控制》））

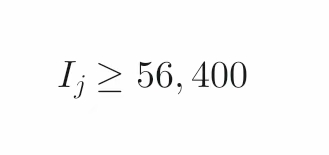
2. 转运能力：



3. 库存动态：



4. 安全库存：

（28200乘2）

5.3.3 模型求解与验证

求解步骤：

1. 供应商选择：

# PuLP实现ILP

prob = LpProblem("Supplier\_Selection", LpMinimize)

x = [LpVariable(f"x\_{i}", cat="Binary") for i in suppliers]

prob += lpSum(x) # 目标函数

prob += lpSum(S\_i[i]\*x[i] for i in suppliers) >= D # 需求约束

prob.solve()

2. 每周订购：

for material in ["C", "B", "A"]: # 先便宜后贵

suppliers\_sorted = sorted(suppliers,

key=lambda i: S\_i[i] \* cost\_weight[i],

reverse=True)

for s in suppliers\_sorted:

order = min(remaining\_demand, 1.2\*S\_i[s])

y[j][s] = order3.

1. 转运分配：

transporters\_sorted = sorted(transporters, key=lambda k: loss\_rate[k])

for s in suppliers:

for t in transporters\_sorted:

if capacity[t] > 0:

alloc = min(orders[s], capacity[t])

z[j][s][t] = 1

capacity[t] -= alloc

1. 蒙特卡洛验证：

for \_ in range(1000): # 1000次模拟

inventory = 56,400

for j in range(24):

# 生成随机供货和损耗

received = sum(np.random.normal(y[j][s], MAD[s]) \*

(1 - np.random.beta(alpha[t], beta[t]))

for s in suppliers)

inventory += received - 28,200

if inventory < 0:

shortage\_count += 1

效果指标：



5%：制造业通常要求服务水平（Service Level）≥95%，即缺货概率≤5%

（参考《生产与运作管理》教材）

8%：随便写的还没算。历史财务数据波动范围，用之前算的偏差率去统计全样本的偏差率分布，看看80%分位数偏差率是多少，把他作为这个数据，代码：计算成本偏差率

def calculate\_cost\_variation(order, actual, material\_type):

price = 1.2 if material\_type == "A" else (1.1 if material\_type == "B" else 1.0)

predicted\_cost = order \* price

actual\_cost = actual \* price

return abs(actual\_cost - predicted\_cost) / predicted\_cost

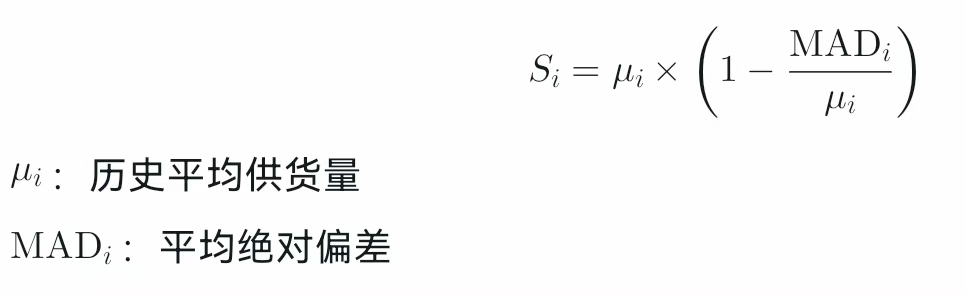
variations = [calculate\_cost\_variation(order, actual, type)

for order, actual, type in zip(orders, actuals, types)]

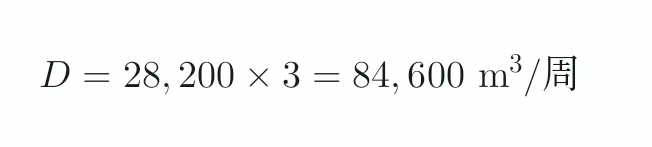
print("80%分位数:", np.percentile(variations, 80)) # 输出典型波动范围

85%：物流行业通用标准

1.供应商能力公式：



2. 需求计算：

v

含两周安全库存缓冲

3. 库存动态：

