MPS成本优化报告

2251912 潘世维

# 问题描述

某企业主要生产产品1与产品2，现需制定为期4个时段的主生产计划（MPS），以最小化总成本。关键要求如下：

* **需求满足**：产品1与产品2的每期毛需求必须被满足。
* **提前期**：均为1个时段。
* **库存初始值**：两产品初始库存均为30件。
* **持有成本**：分别为0.5（产品1）与0.8（产品2）货币单位/件/期。
* **采购成本**：分别为20（产品1）与15（产品2）货币单位/件。
* **最小生产批量**：5件。
* **生产能力**：
  + 产品1在工作中心A与B加工，加工时间分别为2h与3h。
  + 产品2在工作中心A与B加工，加工时间分别为2h与1h。
* **工作中心能力**：
  + A：112小时/期，加班成本2元/小时。
  + B：95小时/期，加班成本2元/小时。

# 成本模型构建

建立了如下成本构成模型：

* **采购成本**：若某时段产出与库存不能满足需求，则需外部采购，按单位采购成本计入总成本。
* **持有成本**：期末存货按单位持有成本累计。
* **准备成本**：每个产品在每期生产时均产生相应工序的准备成本。
* **加班成本**：当工作中心总工时超过标准工时时，超出部分按加班费计入。

所以

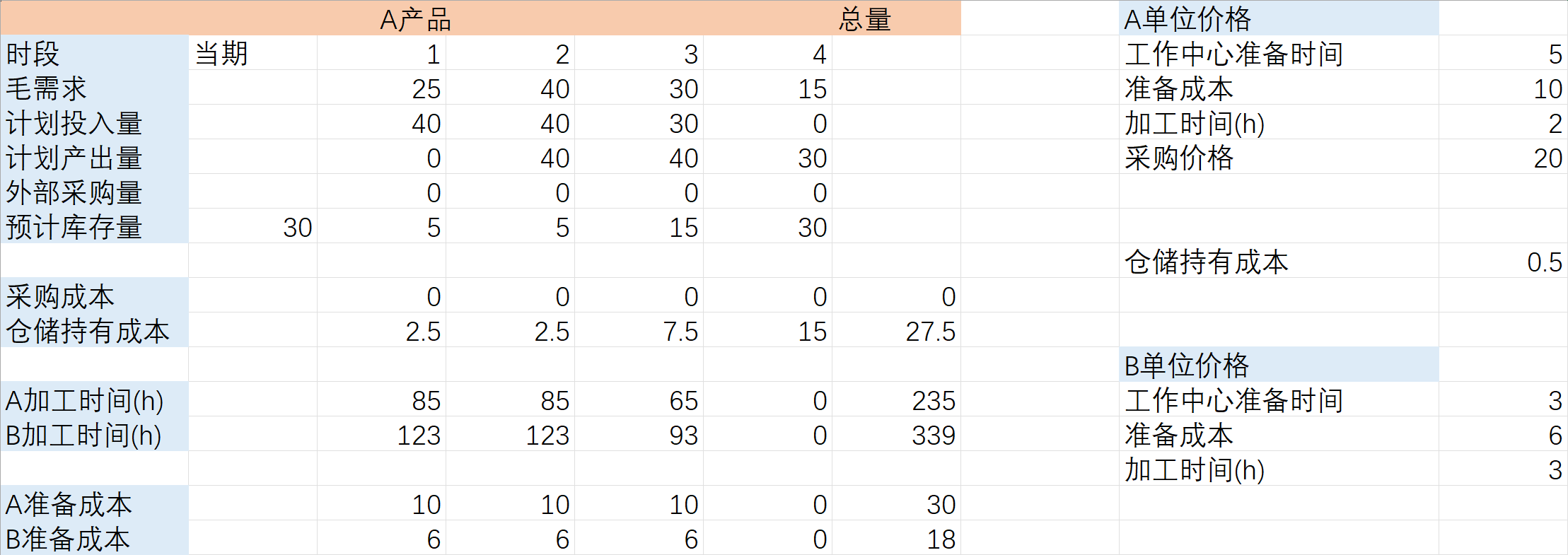
总成本=采购成本+持有成本+准备成本+加班成本

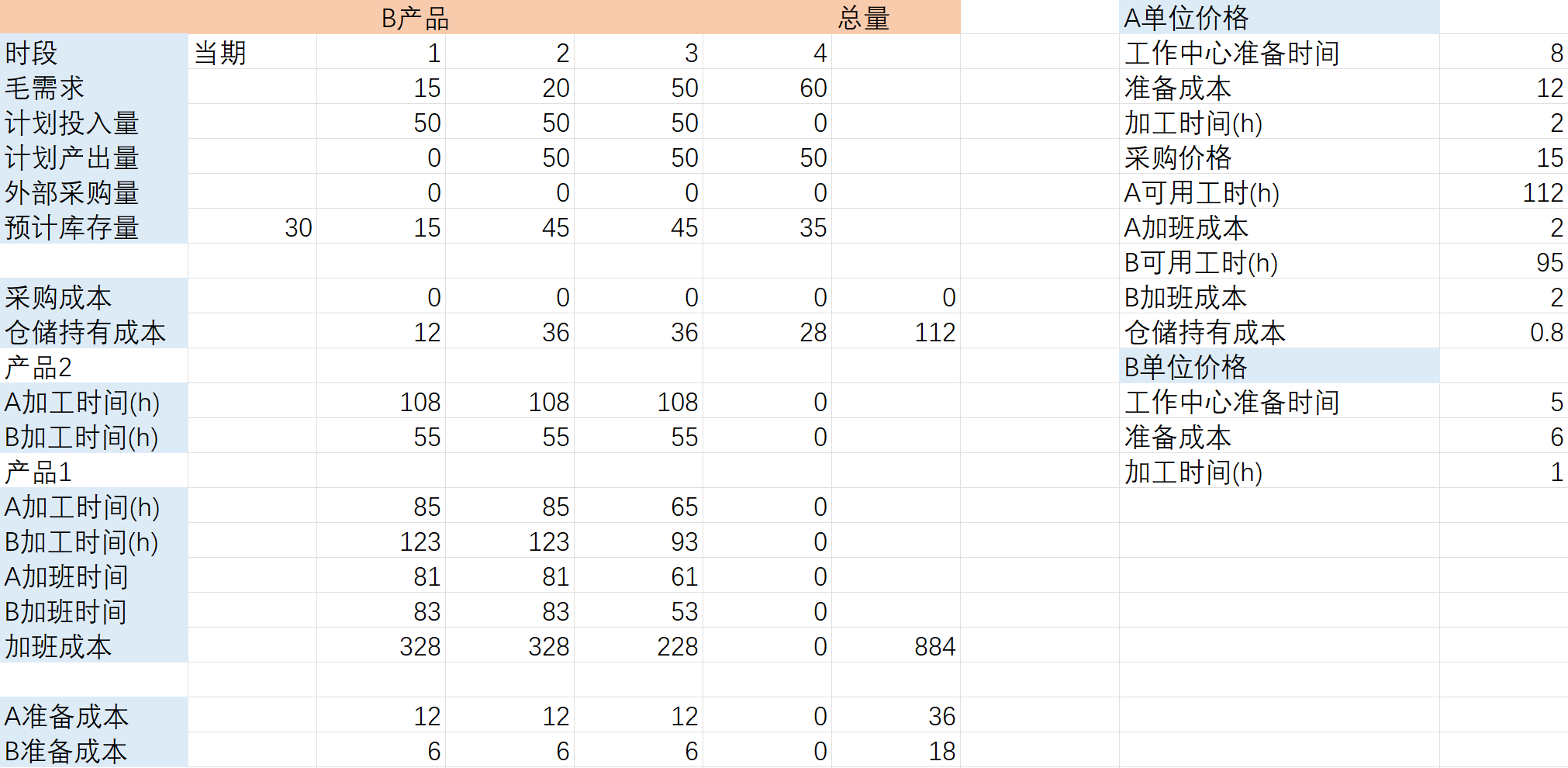
# 计算给定MPS的总成本

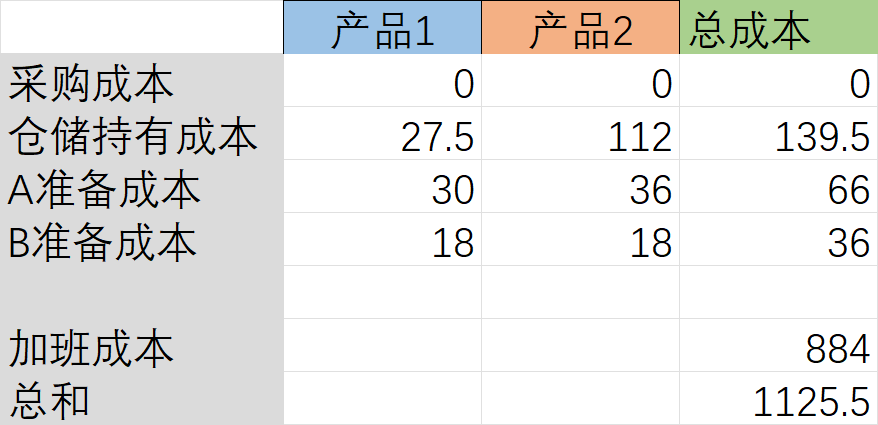
在本节中，我们基于给定的主生产计划（MPS）对整个4个时段的总成本进行计算。给定的计划投入量如下：

* **产品1**：每期的计划投入量为 [40, 40, 30, 0]
* **产品2**：每期的计划投入量为 [50, 50, 50, 0]

用Excel计算如下







用Python代码实现上述计算过程，其中通过calculate\_costs(mps\_plan)函数实现上述逻辑，具体步骤如下：

**步骤1：初始化资源负载与总成本**

center\_hours = {center: [0]\*4 for center in work\_centers}

total\_cost = 0

为每个工作中心记录4个时段的工时使用情况，初始化总成本为0。

**步骤2：逐期处理每种产品**

对于每种产品，在每个时段执行如下操作：

for p\_idx, product in enumerate(products):

    inventory = product.initial\_inventory

    for period in range(4):

        # 如果有生产，则累加加工工时与准备时间

**a. 计算每个工序的总加工时间**

if mps\_plans[p\_idx][period] > 0:

    for op in product.routing:

        hours = op['setup\_time'] + op['process\_time'] \* mps\_plans[p\_idx][period]

        center\_hours[center][period] += hours

每次生产都要重新设置工序，因此准备时间和加工时间都会计入工作中心的负载。

**b. 计算需求缺口与采购成本**

if inventory < gross\_reqs[p\_idx][period]:

    purchase\_qty = gross\_reqs[p\_idx][period] - inventory

    total\_cost += purchase\_qty \* product.purchase\_cost

inventory = 0

若库存不足满足毛需求，需采购缺口部分，并更新总成本。

**c. 更新库存与计算持有成本**

inventory += mps\_plans[p\_idx][period-1] if period > 0 else 0

inventory -= gross\_reqs[p\_idx][period]

total\_cost += inventory \* product.holding\_cost

期末库存用于计算持有成本。

**步骤3：累加准备成本**

if mps\_plans[p\_idx][period] > 0:

    for op in product.routing:

        total\_cost += op['setup\_cost']

只要在该时段生产，就需按产品工序数累加准备费用。

**步骤4：计算每个工作中心的加班成本**

for center\_name, loads in center\_hours.items():

    for load in loads:

        overtime = max(0, load - work\_centers[center\_name].available\_hours)

        total\_cost += overtime \* work\_centers[center\_name].overtime\_cost

检查每期是否超出额定工时，若有则按加班成本进行计费。

得到总成本为1125.5

Given MPS total cost: 1125.50

=== MPS Detailed Metrics ===

Product 1

--------------------------------------------------

Parameter | Period 1 | Period 2 | Period 3 | Period 4

--------------------------------------------------

Gross Req. | 25 | 40 | 30 | 15

Plan Input | 40 | 40 | 30 | 0

Plan Output | 0 | 40 | 40 | 30

Purchase | 0 | 0 | 0 | 0

Inventory | 5 | 5 | 15 | 30

Process Hours | 208.0 | 208.0 | 158.0 | 0.0

Product 2

--------------------------------------------------

Parameter | Period 1 | Period 2 | Period 3 | Period 4

--------------------------------------------------

Gross Req. | 15 | 20 | 50 | 60

Plan Input | 50 | 50 | 50 | 0

Plan Output | 0 | 50 | 50 | 50

Purchase | 0 | 0 | 0 | 0

Inventory | 15 | 45 | 45 | 35

Process Hours | 163.0 | 163.0 | 163.0 | 0.0

Work Center Overtime Hours:

Center A | 81.0 | 81.0 | 61.0 | 0.0

Center B | 83.0 | 83.0 | 53.0 | 0.0

Costs Breakdown:

Setup Cost | 16.0 | 16.0 | 16.0 | 0.0

Purchase Cost | 0.0 | 0.0 | 0.0 | 0.0

Product 2

Holding Cost | 12.0 | 36.0 | 36.0 | 28.0

Setup Cost | 18.0 | 18.0 | 18.0 | 0.0

Purchase Cost | 0.0 | 0.0 | 0.0 | 0.0

Overtime Costs:

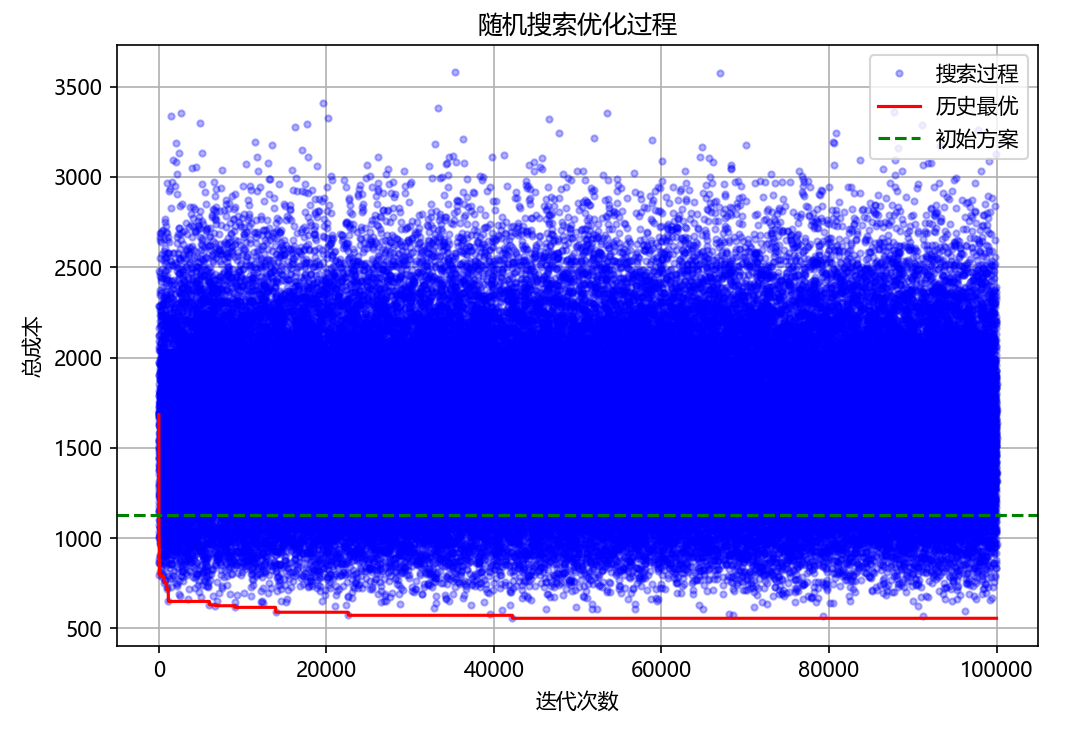
Center A | 162.0 | 162.0 | 122.0 | 0.0

Center B | 166.0 | 166.0 | 106.0 | 0.0

和Excel计算结果相同，证明结果正确。

# 搜索算法设计与优化

首先，用最简单的纯随机搜索，设置100000次迭代。



优化过程如下图所示：

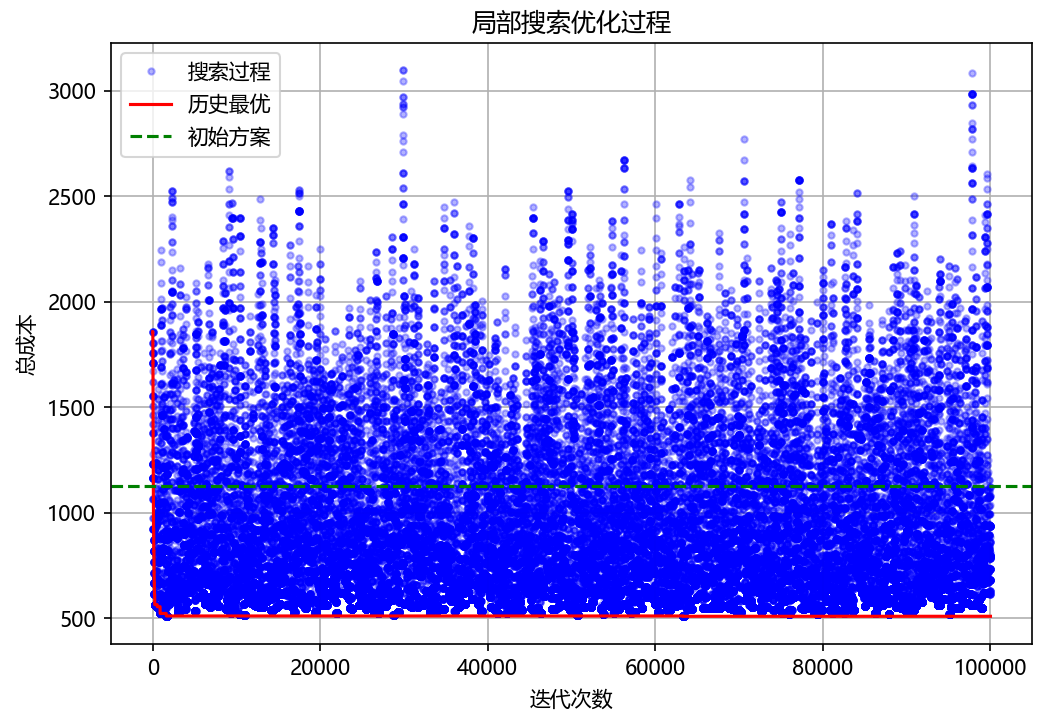
* 蓝色点为每次随机生成方案的成本；
* 红色线为当前最优成本的逐步演化；
* 绿色虚线为原始MPS成本（1125.5）；

纯随机搜索结束后，返回的当前最优解如下：

* **产品1**：每期的计划投入量为 [35, 30, 15, 10]
* **产品2**：每期的计划投入量为 [30, 40, 45, 30]

最优总成本为556.5

再使用局部搜索+ 重启方法



返回的当前最优解如下：

* **产品1**：每期的计划投入量为 [40, 25, 15, 0]
* **产品2**：每期的计划投入量为 [10, 60, 45, 0]

最优总成本为509，得到了明显提升

# 结论与建议

本次研究通过建立精确的成本模型并利用Python模拟验证，实现了主生产计划（MPS）在满足多重约束下的**最小化总成本优化**目标。总结如下：

* **模型有效性已验证**：计算结果与Excel一致，保障了算法的可靠性；
* **搜索算法效果显著**：从初始1125.5降至509，节约超54%成本；
* **方法通用性强**：该方法框架可拓展至多品种、多期、柔性工序生产计划场景；
* **建议**：企业在制定MPS时，可结合本方法进行成本敏感性分析及参数调优，提升排产效率与成本控制能力。