基于多种复杂约束条件下的智能排产

摘要

在实际的工程塑料生产问题中，生产线分配受到许多约束条件的限制。我们在问题一中根据问题的约束条件建立了状态转移方程。结合贪心策略和广度优先搜索方法确定了问题的可行解。在问题二中我们考虑了订单超期成本，以及订单换产成本的影响，建立了改进后的规划模型并求解得到了可行解。

问题一仅要求尽可能保证客户交期，这一目标可以具体化为让延期交付订单数量最少。在确定了问题的约束条件和目标函数后，我们可以建立一个生产线的状态（包括总计完成的订单和正在加工的订单）随时间改变的状态转移模型来模拟生产线生产过程。并依据问题的约束条件来影响状态方程和辅助向量的更新过程。

在问题一的求解过程中，我们首先对数据进行了预处理，排除了无法按照约束条件生产的部分订单（例如只给出了大机器上产能的小订单）。之后，我们在约束条件随机选取问题的初始状态，并使用包含剪枝策略的，时间复杂度为 $O(n^4)$ 的广度优先搜索算法来搜索问题后继的可能情况，实现对问题的求解。求解得到的最好情况下有 6 个延期交付的订单。

在问题二中，我们需要考虑订单超期成本，以及订单的综合换产成本。其中方案的总订单超期成本可以用超期订单所对应的订单数的总和来衡量。同时在涉及转产时依据基料换产矩阵来确定订单的基料转产成本，并给出在基料转产成本和填料以及颜色转产成本的比值不同的情况下的总转产成本，并构建使得总成本最小的目标函数。

在问题二的求解过程中，在确定了模型与问题一中的模型在目标函数上的差异后，我们可以采用与问题一相似的状态转移方法来对这一规划模型求解。由于对目标函数的修改，我们改进后的模型在这里更加注重较大订单的完成，得到的最小成本为 0.091。

模型通过建立模拟生产过程的状态转移模型的方法，合理的分析了在生产安排过程中的各种约束条件对于状态转移的影响，并使用搜索的方式确定了不同情境下的可行解。方法在中等问题规模的多约束排产问题中有一定的应用价值。

关键字：随机模拟　状态转移　广度优先搜索

一、 问题重述

工厂可生产具有不同颜色、基料、填料、填料比例等属性的若干种型号的工程塑料产品，这些属性影响着不同型号产品之间的换产成本。厂内共有 4 条高产能大产线和 3 条低产能小产线。订单生产时主要存在着如下规则：

1. 大订单大机台约束：订单数量为 5000 以上的为大订单，大订单如果能在大机台生产，那么不在小机台生产。
2. 小订单跟单约束：订单数量为 2000 以下的为小订单，非本色和白色的小订单的基料、颜色、填料种类、填料比例四个属性与前面的大订单完全一致时，才能在大订单后面跟单生产
3. 订单的相应原料到货之后才能开始生产
4. 5、7 号线为辅助线，本色、白色的订单应尽可能在 5、7 号线安排
5. 若该产线前一订单的产品型号与当前订单的不同，则需要两个小时换洗时间

首先需要考虑问题的简化情况，在不考虑换产成本，仅以保证客户交期为目标的前提下，建立此问题的数学模型并设计相应的排产算法。并利用附件中的订单数据检验模型和算法。

在实际情况中，订单超期有一定的惩罚或者成本，订单超期成本通常以订单超期天数来衡量。此外，实际生产情况中也存在换产成本，关于换产成本有如下规则：

- 基料换产成本 > 颜色换产成本 > 填料换产成本；
- 因为基料不一样换洗后，接下来连续生产的所有同基料的订单颜色应该由浅色往深色排；
- 基料和颜色都一样的订单排在一起时，填料为“无”的订单优先排在前面；
- 当基料、颜色、填料都一样的订单排在一起时，填料比例优先从低到高排。

在综合考虑订单超期成本和各属性的综合换产成本后，在之前分析的基础上建立此问题的数学模型并设计合理有效的排产算法。并利用附件中的订单数据检验模型和算法。

二、 问题分析

问题来源于实际的工程塑料生产问题，问题中基于实际生产中受到的限制设置了许多约束条件，我们把题目中给出的约束条件整理如下：

1. 大订单大机台约束：订单数量为 5000 以上的为大订单，大订单如果能在大机台生产，那么不在小机台生产

2. 小订单跟单约束：订单数量为 2000 以下的为小订单，非本色和白色的小订单的基料、颜色、填料、填料比例四个属性与前面的大订单完全一致时，才能在大订单后面跟单生产
3. 订单的相应原料到货之后才能开始生产
4. 5、7 号线为辅助线，本色、白色的订单应尽可能在 5、7 号线安排

问题一仅要求尽可能保证客户交期，这一目标可以具体化为让延期交付订单数量最少。在确定了问题的约束条件和目标函数后后，我们可以建立一个数学规划模型。由于建立的数学规划模型包括较多的条件判断以及未精确描述的部分，不易给出问题的精确形式。我们可以考虑建立状态转移模型辅助进行求解。我们把所有生产线总计完成的订单和各条生产线正在加工的订单作为订单的状态，根据问题的约束条件给出了状态转移方法和问题受到的其他约束条件，从而模拟各条流水线的生产过程。

在问题一的求解过程中，我们首先对数据进行了预处理，排除了无法按照约束条件生产的部分订单（例如只给出了大机器上产能的小订单）。之后，我们可以通过随机选择可行的开始状态，并对各种可能状态进行广度优先搜索的方法来求解状态转移模型并得到可行解。

在问题二中，我们需要考虑订单超期成本，以及订单的综合换产成本。其中方案的总订单超期成本可以用超期订单的订单数量的总和来衡量。同时在涉及转产时依据基料换产矩阵来确定订单的转产成本。我们可以在问题一已经建立的模型的基础上增加使得总成本最小的目标函数，建立考虑了订单超期成本和订单换产成本的数学规划模型。在规划模型确定后，我们可以采用与问题一相似的方法求可行解。

三、 模型假设及符号说明

3.1 模型假设

- 生产时间以小时为单位，即每条生产线每个小时最多只进行一个订单的生产工作
- 认为订单数量为 2000 至 5000 的为中等订单，这类订单可以在两类产线上生产
- 生产开始时间假定为 8 月 1 日 07：00；原料到货时间假定为当天 07：00；客户交期假定为当天 00：00
- 一个订单能且仅能在一条生产线上一次性完成

3.2 符号说明

符号	符号说明	备注
t	距离 8 月 1 日 7 点的小时数	
$l_{(t,i)}$	时刻 t 第 i 条生产线正在生产的订单的编号	(i=1,2,…,7)
$L_{(t,i)}$	表示时刻 t 第 i 条生产线可以生产的订单的集合	同上
$\tau_{(t,i)}$	表示在时刻 t 第 i 条生产线生产完当前订单的剩余时间	同上
α_j	第 j 个订单的属性向量	

四、 问题一

4.1 模型准备

问题一是一个以逾期订单数量最少为目标的优化问题。问题以实际的工程塑料生产为背景，涉及了在实际工业生产的排产问题中的多种约束条件。在建立模型前,我们首先需要对数据进行清洗以去除部分没有考虑价值的数据，同时对该生产线的约束条件进行一定的分析，并且给出模型中涉及的部分状态符号的定义与数学表示。

4.1.1 数据清洗

由于问题的数据直接来源于生产实际，数据中包含了部分冗余数据。在数据清洗部分中，我们首先去除了部分没有订单需求，但仍然给出了机器产能的冗余数据。之后，考虑到某些型号的产品只能在特定的大机台生产，而小订单只能通过跟单的方式在大机台生产。因此我们去除了部分没有参考意义的无法安排生产的小订单订单数据。在完成数据清洗后，一共剩余 61 组有效订单。

4.1.2 约束说明

在实际的生产中，根据订单的规模和产品的不同，可供选择的流水线也有一定的差异。问题的约束条件整理如下：

1. 大订单大机台约束：订单数量为 5000 以上的为大订单，要求大订单如果能在大机台（1、2、3、5 号线）生产，那么不在小机台（4、6、7）生产；
2. 小订单跟单约束：订单数量为 2000 以下的为小订单，非本色和白色的小订单应尽量在小机台生产，如果在大机台生产，只有当小订单的基料、颜色、填料、填料比例四个属性与前面的大订单完全一致时，才能跟单大订单生产。

3. 特殊订单约束：本色、白色的订单，尽可能排在 5、7 号线
4. 小订单超期约束：小订单可以超期一天
5. 原料约束：订单的相应原料到货之后才能开始生产
6. 换洗时间约束：生产某订单时，若该产线前一订单的产品型号与当前订单的不同，则需要两个小时换洗时间。

4.1.3 符号说明

在建立模型前，我们需要一些向量来表示各条生产线的状态以及可供生产线生产的订单。生产线的生产状态可以由生产线正在生产的订单和所有生产线已经完成的订单来表示。为了反映某一时刻生产线的所有生产线的生产状态，我们建立下面的 8 维状态向量:

$$\Omega_t = [\sigma_t, l_{(t,1)}, l_{(t,2)}, l_{(t,3)}, l_{(t,4)}, l_{(t,5)}, l_{(t,6)}, l_{(t,7)}]^T \tag{1}$$

其中 t 表示当前的时间（距离 8 月 1 日 7 点经过的小时数），$l_{(t,i)}(i = 1, 2, \cdots, 7)$ 表示在时刻 t 第 i 条生产线正在生产的订单的编号，σ 的值表示**当前时刻 t 已经完成**的订单的数量。

由于在实际情况中生产线的生产受到各种约束条件的限制。为了表示各条生产线可生产的订单，我们建立以下 7 个辅助集合:

$$L_{t1}, L_{t2}, L_{t3}, L_{t4}, L_{t5}, L_{t6}, L_{t7} \tag{2}$$

其中 $L_{ti}(i = 1, 2, \cdots, 7)$ 表示时刻 t 时第 i 条生产线可以生产的订单的集合。再把集合 L_{ti} 划分为 L_{tib}，L_{tim} 和 L_{tis}，分别表示时刻 t 第 i 条生产线可以生产的大订单的集合，中等订单（订单数量为 2000-5000）的集合和小订单的集合。

在生产过程中，每条产线可能处于生产状态（甘特图的蓝色部分）。也可能由于完成生产或原料不足等因素处于暂停状态（甘特图的白色部分），如下图所示:

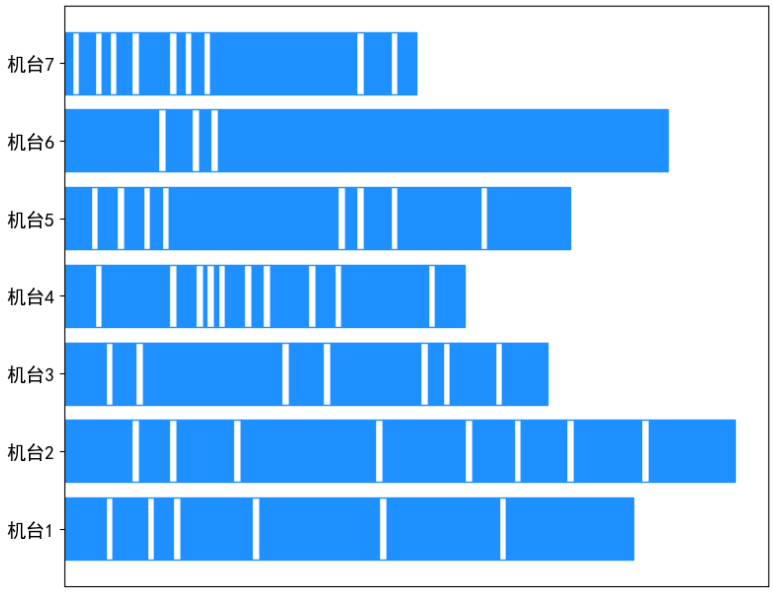


图 1　关于整个生产项目的甘特图

为了表达各条生产线的具体生产情况，我们建立下面的时间辅助向量：

$$\beta_t = [\tau_{t1}, \tau_{t2}, \tau_{t3}, \tau_{t4}, \tau_{t5}, \tau_{t6}, \tau_{t7}]^T \tag{3}$$

其中 $\tau_{ti}(i = 1, 2, \cdots, 7)$ 表示在时刻 t 第 i 条生产线生产完当前订单所需要的**剩余时间**，当 $\tau_{ti} = 0$ 时，表示时刻 t 第 i 条生产线处于空置的状态。

在分配生产时，为订单分配的生产线受到订单的产品型号、颜色、订单数量等因素的影响。为了反应各个订单的属性，我们使用多维向量 α_i 表示第 i 个订单：

$$\alpha_i = [a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i]^T, \quad (i = 1, 2, \cdots, 65) \tag{4}$$

其中 $a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i$ 分别表示第 i 个订单从产品型号到填料比的 8 个属性值，分别为：产品型号、原料到货时间、颜色、订单数量、客户交期、基料、填料、填料比例。

4.2 状态转移模型建立

基于上面的分析，我们可以通过状态向量 Ω_t 和辅助向量 β_t 来确定时刻 t 生产线的生产状态。而各个订单的属性 α_i 又决定了生产线可以生产的订单 L_{ti}。在此基础上，我们可以建立描述生产线情况的状态转移模型。

4.2.1 状态初值选择

在通过状态转移模型模拟生产过程之前，我们需要确定生产线初始生产的订单，也就是状态转移模型的状态初值。我们可以使用随机化算法生成状态初值，枚举出所有可能的初值并分别从这些初值进行状态转移，直到 $t = t_{end}$。但由于问题包含的订单的可能情况较多，而且直接搜索所有可行解的时间复杂度约为 $O(n^8)$，直接进行随机生成并遍历可行情况的方法计算量过大。我们需要对搜索过程进行合理的剪枝来降低时间开销。

我们让初值 $t = 0$，我们可以把 l_i 限制在对应的 L_i 的范围内，并且满足条件约束的情况下，随机选择订单 $L(0, i)$，来产生初值并且添加到初始队列中。这种策略排除了订单与生产线不匹配的情况，可以提高状态搜索的效率。

4.2.2 状态向量的更新

随着时间 t 的推移，状态向量随着时刻 t 进行更新，每次步进时间间隔为 Δt_i，有两种可能的情况会使得时间步进后状态发生改变，一种情况是时刻 $t + \Delta t_i$ 时，某一订单原料到货，原来由于没有可生产订单而空闲的产线开始生产该订单；另一种情况是一条产线订单生产完毕，总订单完成量 $\sigma_{t+\Delta t_i}$ 加一，并且需要决定接下来继续生产的订单，下面对两种情况分别进行讨论。

在某一订单原料到货情况下的状态转移 假设时刻 t 步进 Δt_i 后，订单 p 的原材料到货，机台 $l_{(t+\Delta t_i,i)} = l_{(t+\Delta t_i,j)} = l_{(t+\Delta t_i,k)} = \cdots = 0$ ，表示在时刻 $t + \Delta t_i$，机台 $i, j, k \cdots\cdots$ 处于闲置状态。并且机台 i, j, k 可以用于生产订单 p，即:

$$p \in L_{(t+\Delta t_i,i)}, p \in L_{(t+\Delta t_i,j)}, p \in L_{(t+\Delta t_i,j)}, \cdots$$

那么状态由 Ω_t 转移到 $\Omega_{t+\Delta t_i}(l_{(t+\Delta t_i,i)} = p)$ 或 $\Omega_{t+\Delta t_i}(l_{(t+\Delta t_i,j)} = p)$ 或 $\Omega_{t+\Delta t_i}$ $(l_{(t+\Delta t_i,k)} = p) \ldots\ldots$ 同时把这些状态以此添加到队列的末尾。

某一订单生产完毕情况下的状态转移 假设在 t 时刻第 i 条产线完成生产，令:

$$\tau_{min} = \min \beta_t = \min \tau_{ti}, (i = 1, 2, \cdots, 7)$$

则状态 Ω_t 转移到状态 $\Omega_{t+\tau_{min}}$。

为方便叙述，我们假设 $\tau_{min} = \tau_{t1}$，即第 1 条产线最先完成订单的生产，假设:

$$i \in L_{(t+\Delta t_i,1)}, j \in L_{(t+\Delta t_i,1)}, k \in L_{(t+\Delta t_i,1)}, \cdots$$

那么状态由 Ω_t 可以转移到 $\Omega_{t+\Delta t_i}(\sigma + 1, l_{(t+\Delta t_i,1)} = i)$ 或 $\Omega_{t+\Delta t_i}(\sigma + 1, l_{(t+\Delta t_i,1)} = j)$ 或 $\Omega_{t+\Delta t_i}(\sigma + 1, l_{(t+\Delta t_i,1)} = k)$。

4.2.3 辅助集合的更新

在时刻 t，每条产线的当前时刻 t 可生产订单有三种情况可能会出现更新，依次为当订单原材料到货时，当订单被产线生产时，当订单逾期时，下面分别讨论如下。

当时刻 t 订单 i 的原材料到货时，如果某条产线 j 可以生产订单 i，订单 i 就成为了产线 j 的可生产订单，把 i 添加到 L_{tj} 中。

当时刻 t 订单 i 被生产时，应当把已经生产的订单移除。如果某条产线 j，$i \in L_{tj}$，则从 L_{tj} 中删掉 i。

当时刻 t 订单 i 逾期时，应当把已经逾期的订单移除。如果某条产线 j，$i \in L_{tj}$，则从 L_{tj} 中删掉 i。

4.2.4 时间辅助向量的更新

每经过 Δt_i 的时间，正在生产的生产线的剩余生产时间减少 Δ_t，而未生产的生产线的剩余时间不变。可以让 $\beta_{t+\Delta t_i}$ 在 β_t 的基础上按如下规则进行更新：

$$\begin{cases} \tau_{t+\Delta t_i} = \tau_t - \Delta t_i, (\tau_t \neq 0) \\ \tau_{t+\Delta t_i} = 0, (\tau_t = 0) \end{cases} \tag{5}$$

4.3 状态转移模型的约束条件

在明确了状态转移模型后，我们可以把4.1.2提及的约束条件进行如下的数学表示：

1. 大订单大机台约束 订单 5000 以上为大订单，大订单只能在大机台上生产。假设订单 i，并且有 $d_i > 5000$，那么在更新辅助集合时，只能在时刻 t 将 i 添加到 $L_{tj}, (j = 1, 2, 3, 5)$。

2. 小订单跟单约束 非本色和白色的小订单必须跟在大订单后面才能在大机台生产。表示为如果 $l_{ti}(i = 1, 2, 3, 5) = j(d_j < 2000)$，那么一定有 $l_{(t-\Delta t_i, i)}(i = 1, 2, 3, 5) = k(d_k > 5000)$，且 $a_j = a_k$。

3. 特殊订单约束 本色和白色的订单只在 5 号线和 7 号线生产。如果订单 α_j 的颜色属性 o_j 表示本色和白色，即 $o_i = 1$，则只把 j 添加到 L_{t5} 或者 L_{t7} 中。

4. 小订单超期约束 小订单可以超期一天。对于订单 i，并且有订单数 $d_i < 2000$，那么订单的逾期时间 f_i 增加 24 小时。

5. 换洗时间约束 生产某订单时，若该产线前一订单的产品型号与当前订单的不同，则需要两个小时换洗时间。在进行的状态转移时，某条产线上订单 i 在时刻 t 还需经过 Δt_i 的时间完成，接下来生产订单 j，如果两订单的产品型号相同，即 $a_i = a_j$，那么状态由 Ω_t 转移到 $\Omega_{t+\Delta t_i}$，否则需要两个小时的换洗时间，状态由 Ω_t 转移到 $\Omega_{t+\Delta t_i+2}$。

4.4 终止状态

我们使用广度优先搜索来搜索状态，当某一状态的总时长超过了终止时长 t_{end}，即 $t \geq t_{end}$，这代表着加工的最后期限已经到达，因此我们可以把 Ω_t 视为一个终止状态，终止状态不再产生后继状态，并从队列中取出，添加到终止状态集合 S_Ω 中。当所有的状态都到达了终止状态，视为搜索结束。选择终止状态集合 S_Ω 中 σ 的值最大，即完成订单最多的的终止状态，进行回溯，即可得到完成订单数最多的一种可行方案。

4.5 状态回溯

我们使用一个变量 num 记录当前产生的总状态数，并且把 num 作为当前最新产生状态的标记，我们在由状态 num 生成新状态 $num+1, num+2, \cdots$ 时，使用数组 p 记录新状态的“上一个状态”，即有 $p_{num+1} = num, p_{num+2} = num, \cdots$。我们在进行步骤 5 时，选择一个终止状态一直向上回溯，直到初始状态，然后对这些状态的含义进行解析，即得到完成订单数最多的方案。

4.6 模型求解

我们使用宽度优先搜索算法从初值搜索状态值，使用一个队列保存还没有被搜索到的值。每次从队列头部取出一个状态值，把这个状态值所有能拓展到的下一个状态添加到队列的尾部。以此类推知道状态达到终点，从而找到所有达到终点的状态，然后比较这些状态完成订单数的多少。找到完成订单数最多的一种状态，通过回溯法找到所有的中间状态，还原订单生产的整个过程。算法的具体流程如下：

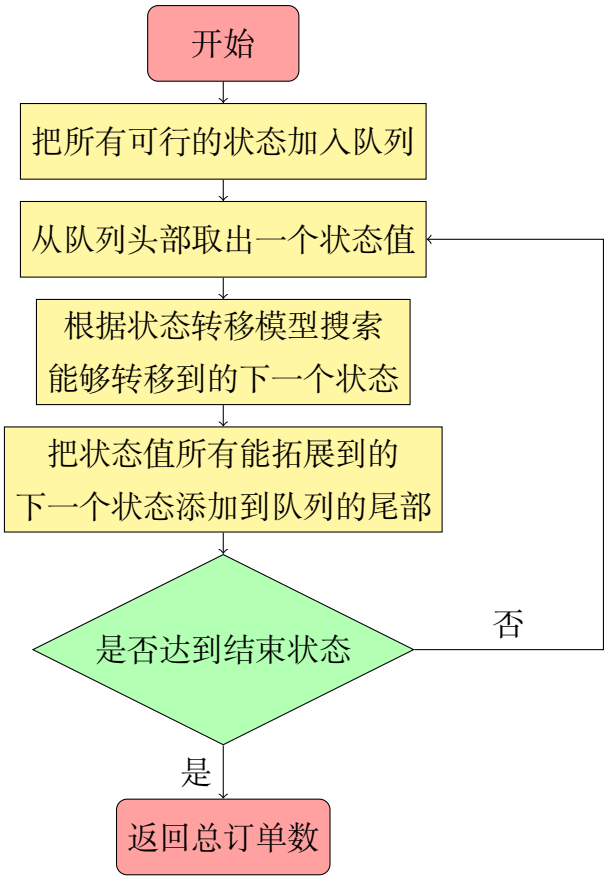
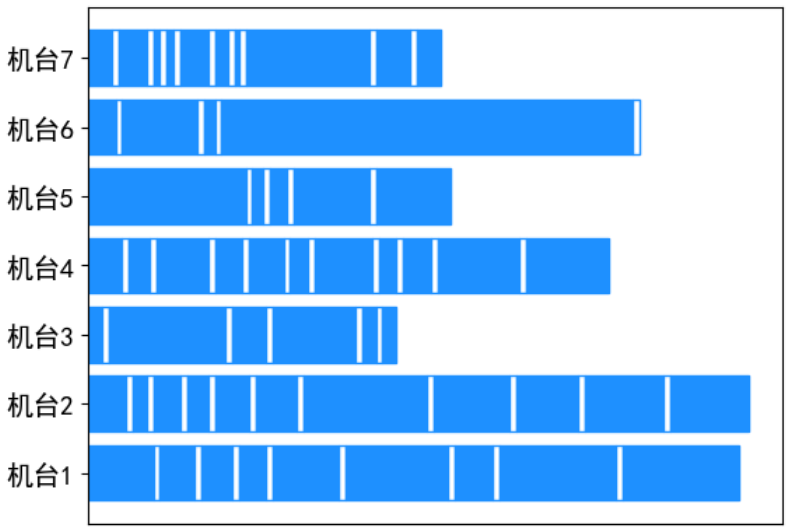


图 2　搜索算法的流程图

通过上述算法，我们找到的能够使得完成订单数最多的方案能够完成 55 组订单。能完成 55 组订单的一种情况的甘特图如下（白色代表空闲时段和因换洗停止生产的时段）：



4.6.1 算法分析

我们团队的算法的时间复杂度以及算法的有效性分析如下：

不妨设订单总数为 n，订单产品量均值为 a，总生产时间为 t，迭代次数为 m。算法在搜索过程中沿着时间轴在上面的几个维度上进行搜索，时间复杂度约为约为 $O(namt)$。在完全不使用剪枝策略时算法的时间复杂度较大。

算法通过建立状态转移模型来模拟生产线的生产过程。在4.3节综合考虑了大订单、小订单、特殊订单、换洗时间等约束条件，并通过这些约束条件影响订单的状态转移，较好的反映了订单生产的全过程。

4.6.2 算法优化

空间优化——时间离散化处理 题目中给出的订单生产时间从 8 月 1 日 7:00 开始，到 8 月 9 日 0:00 截止，一共有 185 个小时，但是并不是每个小时都会发生状态改变，因此我们不需要 $\Omega_0, \Omega_1, \Omega_2, \cdots, \Omega_{185}$，我们只需要在状态发生的时刻生成状态表示向量 Ω_t。同理 L_{ti}, β_t 也可以使用相似的离散化处理方法，这样可以降低计算的空间复杂度。

时间优化——剪枝策略 由于问题一只需要考虑，在加入型号改变对于增加的换洗时间对于完工时间的影响的影响后，使用贪心策略可以得到最优情况. 因此，我们可以基于贪心策略以及题目给出的约束条件进行剪枝来优化算法的时间时间复杂度。主要有以下几种优化方案：

1. 在选择开始状态时，把 l_i 限制在对应的 L_i 的范围内，并且满足条件约束的情况下，随机选择订单 L(0, i)，以排除生产线与订单不匹配的情况。
2. 在进行状态转移时优先考虑距离完工时间较近（考虑切换时潜在的换洗时间）的订单

五、 问题二

5.1 模型建立

在问题二中，我们需要综合考虑订单超期成本和各属性的综合换产成本，建立此问题的数学模型并设计合理有效的排产算法。我们可以考虑通过列举不同的基料、颜色和填料换产成本来反映在不同情况下转产成本对总成本的影响。并沿用问题一中对于超期惩罚的确定方法。我们可以在此基础上建立使转产成本最小，超期惩罚最小的数学规划模型。

5.1.1 转产成本模型

根据题目中的已知条件，基料换产成本 W_b> 颜色换产成本 W_c> 填料换产成本 W_d。由于我们无法确定颜色和填料的具体的换产成本，在这里我们可以考虑列举不同的颜色换产成本和填料换产成本与基料换产成本的关系，从而确定在不同条件下的转产成本。

为了模拟三个换产成本差别不大的情况，我们选择 $W_b : W_c : W_d = 4 : 3 : 2$;

为了模拟三个换产成本之间有一定差距的情况，我们选择 $W_b : W_c : W_d = 4 : 2 : 1$;

为了模拟三个换产成本差别较大的情况，我们选择 $W_b : W_c : W_d = 15 : 3 : 1$;

由于我们在这里的假设中的各种成本与基料换产成本直接相关，因而可以把总成本最小的优化问题转化为使基料成本最小的优化问题。

5.1.2 超期惩罚模型

我们可以使用与问题一相近的方法，但是考虑到在问题二中较大的订单超期相对小订单超期有更大的影响，我们改为使用无法按时完成的订单所对应的订单数量来表示超期情况。在确定了转产成本和超期惩罚后，由于我们的目标是使得转产成本和超期惩罚最小化，我们可以用 W 来表示样本的总成本，它可以表示为:

$$W = \sum [W_b(i,j) + W_c(i,j) + W_d(i,j)] + w \times \sum (\xi_k) \tag{6}$$

其中 i 代表之前生产的产品型号，j 代表现在的产品型号，ξ_k 代表未完工订单 k 的订单数量，w 为代表转产成本和超期惩罚重要性的常数。

由于问题二与问题一只在目标函数上有区别（前者考虑了转产成本），而模型的约束条件没有明显改变。我们可以建立使 W 最小的数学规划模型。并使用与问题一相似的状态转移方法求解，具体的状态转移模型如下:

1. 用时刻 t 第 i 条生产线正在生产的订单的编号 $l_{(t,i)}$ 和当前时刻 t 已经完成的订单的数量 σ 描述生产线的状态。
2. 在初值选择时，依据生产线可以生产的订单 L_{0i} 进行随机选择。
3. 在在某一订单原料到货或者某一订单生产完成的情况下进行状态转移，空闲的产线选择新订单。
4. 在订单原材料到货，订单被生产或订单逾期时更新可生产订单。
5. 按照时间更新各条产线完成当前订单所需要的剩余时间。
6. 状态转移过程受到大订单大机台约束、小订单跟单约束等约束条件的限制。

5.2 模型求解

在问题给出的数据中，基料由 i 切换为 j 的换产成本由一个 0-100 的分数 $W_0(i,j)$ 表示，我们把这一分数修正如下：

$$W_b(i,j) = 100 - W_0(i,j) \tag{7}$$

修改后的分数从而能更为直接的反映换产成本的高低。同时我们可以根据5.1.1的方式确定颜色和填料的换产成本。

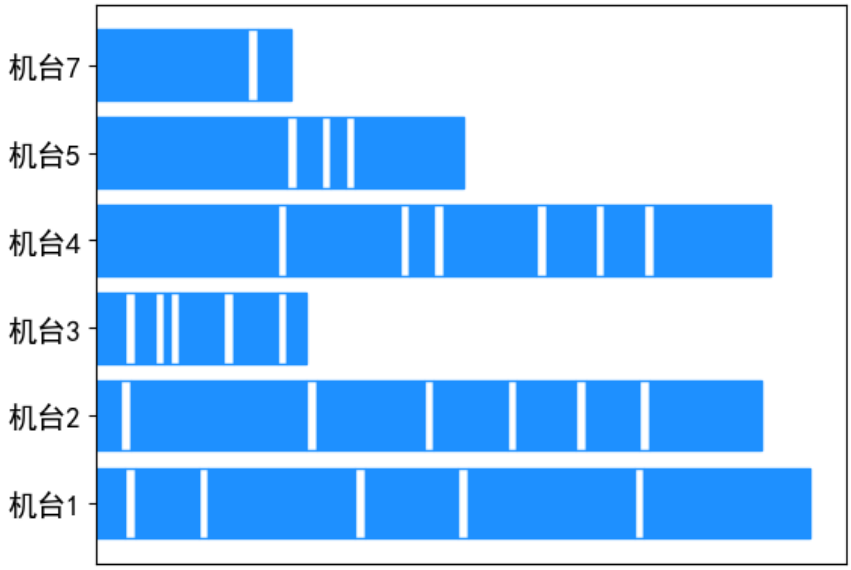
由于我们的转产成本和超期惩罚的均值不同，我们首先需要依据待选择集合的数据来把转产成本和超期惩罚标准化。

在将数据标准化后，我们可以把转产成本和超期惩罚按照 1:1 赋权（即 w=1）。此时目标函数6改写为：

$$W = \sum[W_b(i,j) + W_c(i,j) + W_d(i,j)] + \times \sum(\xi_k) \tag{8}$$

我们可以使用与问题一相似的方法对订单的生产过程进行模拟。在过程中，我们使用使用宽度优先搜索算法从初值搜索状态值，按顺序从各个状态值所能转移到的状态中进行搜索，最终找到完成订单数最多的一种状态。并通过回溯法找到所有的中间状态来还原订单的生产过程。

通过模拟，我们得到的最好情况下订单的转产成本为 0.091，这种安排可以表示如下图：



通过观察我们发现在考虑了转产惩罚并基于订单大小修改了超期成本后，我们的模

型更倾向于选择较大的订单进行生产，而忽视了部分小订单。这是由于在考虑转产惩罚后，优先生产大订单可以有效减少超期成本，从而降低总的生产成本。

六、 模型评价与推广

6.1 模型优点

1. 模型在状态转移模型建立时较好的考虑了各个约束条件对状态向量的影响
2. 模型对于问题二采用了双目标规划模型，能够较好的描述问题的优化目标
3. 模型根据生产模型的内在特性，给出了时间离散化处理、剪枝策略等算法优化方法，能够有效的降低算法的时间复杂度和空间复杂度

6.2 模型缺点

1. 搜索算法的时间复杂度偏高 (约为 $O(n^4)$)，在更大规模的问题上可能不适用
2. 搜索算法的初始状态选取有随机性，可能无法得到最优解
3. 由于缺乏颜色和基料换产成本的相关数据，只能通过给定的特殊比值模拟不同的生产情况

参考文献

[1] 李昊. 多品种小批量混流装配线自适应排产方法研究 [D]. 中北大学,2021.

附录 A　问题一的状态转移模型的 Python 代码

```
import pandas as pd
import numpy as np
import enum
import random
import math

def getStageTime(frame):
    dictlist={'1J0':-1,'2J0':-1,'3J0':-1,'4J0':-1,'5J0':-1,'6J0':-1,'7J0':-1}
    for item in frame.itertuples():
        index=str(item[3])+str(item[8])
        stages=math.ceil(float(item[11])/item[4])
        dict1={index:stages}
        dict2={index+'-1':2}
        dictlist.update(dict1)
        dictlist.update(dict2)
    return dictlist

def getNumDict(frame):
    numdict={}
    numbasdict={}
    for item in frame.itertuples():
        index=item[1]
        num=item[2]
        bas=item[7]
        numdict.update({index: num})
        numbasdict.update({index: bas})
        numdict.update({index + '-1': num})
        numbasdict.update({index + '-1': bas})
    return numdict,numbasdict

def getBasCost(frame):
    basdict={}
    for item in frame.itertuples():
        index=item[1]
        basdict.update({index+frame.columns[1]:item[2]})
        basdict.update({index + frame.columns[2]: item[3]})
        basdict.update({index + frame.columns[3]: item[4]})
        basdict.update({index + frame.columns[4]: item[5]})
        basdict.update({index + frame.columns[5]: item[6]})
        basdict.update({index + frame.columns[6]: item[7]})
        basdict.update({index + frame.columns[7]: item[8]})
        basdict.update({index + frame.columns[8]: item[9]})
        basdict.update({index + frame.columns[9]: item[10]})
        basdict.update({index + frame.columns[10]: item[11]})
```

```
        basdict.update({index + frame.columns[11]: item[12]})
        basdict.update({index + frame.columns[12]: item[13]})
        basdict.update({index + frame.columns[13]: item[14]})
        basdict.update({index + frame.columns[14]: item[15]})
        basdict.update({index + frame.columns[15]: item[16]})
        basdict.update({index + frame.columns[16]: item[17]})
        basdict.update({index + frame.columns[17]: item[18]})
    return basdict

def getLimitTime(frame):
    lstdict={}
    leddict={}
    for item in frame.itertuples():
        index=item[1]
        slim=(item[3].day-pd.Timestamp('2021-08-01 00:00:00').day)*24
        elim=(item[6].day-pd.Timestamp('2021-08-01 00:00:00').day)*24-7
        lstdict.update({index:slim})
        lstdict.update({index+'-1':slim})
        leddict.update({index:elim})
        leddict.update({index+'-1':elim})
    return lstdict,leddict

def getCompltTime(lis,sdict):
    lcnt = []
    for i in range(1,8):
        lcnt.append(sdict[str(i) + lis[i-1]])
    return lcnt

def unequal(lis1,lis2):
    for i in range(len(lis1)):
        if lis1[i]==lis2[i]:
            return False
    return True

def generateChosen(curlist,avalist,index,t,stdict,eddict,sdict,numdict):
    cand=[]
    litord=['J1','J2','J3','J4','J5','J6','J7','J8','J9','J10','J11','J12','J13','J14','J15','J16',
    'J17','J18','J19','J20','J21','J22','J23','J24','J25','J26','J27','J28','J29']
    for item in avalist[index]:
        if curlist[index]!='J0' and numdict[item]!=numdict[curlist[index]]:
            if item in litord and stdict[item]<=t and
                eddict[item]+24>=t+sdict[str(index+1)+item]+2:
                cand.append(item)
            elif stdict[item]<=t and eddict[item]>=t+sdict[str(index+1)+item]+2:
                cand.append(item)
        elif curlist[index]!='J0' and numdict[item]==numdict[curlist[index]]:
            if item in litord and stdict[item]<=t and
```

```
                eddict[item]+24>=t+sdict[str(index+1)+item]:
                cand.append(item)
            elif stdict[item]<=t and eddict[item]>=t+sdict[str(index+1)+item]:
                cand.append(item)
    #cand = [n for n in avalist[index] if (stdict[n] <= t and eddict[n] >= t + sdict[str(index
        + 1) + n])]
    #print(cand)
    if '-1' in curlist[index]:
        curlist[index]=curlist[index].replace('-1','')
    elif len(cand)==0:
        curlist[index]='J0'
    else:
        chosn=cand[random.randint(0,len(cand)-1)]
        while (chosn in curlist) or (chosn+'-1' in curlist):
            chosn=cand[random.randint(0,len(cand)-1)]
        if curlist[index]!='J0' and numdict[chosn]!=numdict[curlist[index]]:
            chosn=chosn+'-1'
        curlist[index]=chosn
    return curlist

def removal(avalist,item):
    if item in avalist[0]: avalist[0].remove(item)
    if item in avalist[1]: avalist[1].remove(item)
    if item in avalist[2]: avalist[2].remove(item)
    if item in avalist[3]: avalist[3].remove(item)
    if item in avalist[4]: avalist[4].remove(item)
    if item in avalist[5]: avalist[5].remove(item)
    if item in avalist[6]: avalist[6].remove(item)

    if '-1' in item:
        tempitem=item.replace('-1','')
        if tempitem in avalist[0]: avalist[0].remove(tempitem)
        if tempitem in avalist[1]: avalist[1].remove(tempitem)
        if tempitem in avalist[2]: avalist[2].remove(tempitem)
        if tempitem in avalist[3]: avalist[3].remove(tempitem)
        if tempitem in avalist[4]: avalist[4].remove(tempitem)
        if tempitem in avalist[5]: avalist[5].remove(tempitem)
        if tempitem in avalist[6]: avalist[6].remove(tempitem)
    return avalist

class stat:
    def __init__(self,lorder,ccnt,t):
        self.lorder=lorder
        self.ccnt=ccnt
        self.t=t

productFrame=pd.read_excel('product.xls')
```

```
producerFrame=pd.read_excel('producer.xls')
basmc=pd.read_excel('basmcost.xls')

namelist=productFrame['产品型号'].tolist()

#touseFrame=producerFrame[producerFrame['产品型号'].isin(namelist)]
touseFrame=pd.read_excel('touseprod.xls')

usedFrame=touseFrame.merge(productFrame)

basdict=getBasCost(basmc)
sdict=getStageTime(usedFrame)
stdict,eddict=getLimitTime(productFrame)
numdict,numbasdict=getNumDict(productFrame)

avalist=[['J6','J10','J16','J22','J27','J36','J38','J39','J41','J46','J52','J53','J56',
'J59','J60','J62','J63','J64'],
        ['J1','J4','J7','J11','J19','J23','J26','J31','J32','J35','J38','J42','J43',
        'J44','J46','J48','J49','J52','J54','J56','J57','J58','J61','J62','J65'],
        ['J4','J7','J8','J11','J12','J14','J26','J27','J31','J40','J41','J43','J44',
        'J45','J46','J56','J58','J62'],
        ['J3','J4','J5','J7','J11','J12','J14','J19','J20','J23','J26','J28','J29',
        'J32','J35','J37','J38','J42','J43','J45','J46','J48','J52','J56','J60'],
        ['J6','J11','J13','J15','J16','J17','J18','J21','J25','J27','J28','J30','J33',
        'J34','J35','J37','J41','J46','J51','J56','J60','J62'],
        ['J9','J12','J24','J48','J51','J55'],
        ['J2','J3','J6','J11','J13','J17','J18','J19','J21','J25','J30','J33',
        'J34','J46','J47','J50','J56','J60']]

maxtot=0
mintot=100
maxcost=0
mincost=100
finlist=[]
target=10000

for ite in range(4000):
    avalistsuccess=[['J10','J27','J36','J38','J39','J41','J46','J52','J53','J56','J59'
    ,'J60','J62','J63','J64'],
            ['J11','J31','J32','J35','J38','J42','J43','J44','J46','J48','J49','J52',
            'J54','J56','J57','J58','J61','J62','J65'],
            ['J8','J11','J12','J27','J31','J40','J41','J43','J44','J45','J46','J56',
            'J58','J62'],
            ['J3','J4','J5','J7','J11','J12','J14','J19','J20','J23','J26','J28',
            'J29','J32','J35','J37','J38','J42','J43','J45','J46','J48','J52','J56','J60'],
            ['J11','J27','J30','J33','J34','J35','J37','J41','J46','J51','J56','J60','J62'],
            ['J9','J12','J24','J48','J51','J55'],
```

```
        ['J2','J3','J6','J11','J13','J17','J18','J19','J21','J25','J30','J33'
        ,'J34','J46','J47','J50','J56','J60']]

cand=[[n for n in avalistsuccess[0] if stdict[n]==0],
       [n for n in avalistsuccess[1] if stdict[n]==0],
       [n for n in avalistsuccess[2] if stdict[n]==0],
       [n for n in avalistsuccess[3] if stdict[n]==0],
       [n for n in avalistsuccess[4] if stdict[n]==0],
       [n for n in avalistsuccess[5] if stdict[n]==0],
       [n for n in avalistsuccess[6] if stdict[n]==0]]

chosn=[]
cnt=0
while cnt<7:
    item=cand[cnt][random.randint(0,len(cand[cnt])-1)]
    while item in chosn:
        item=cand[cnt][random.randint(0,len(cand[cnt])-1)]
    chosn.append(item)
    removal(avalistsuccess,item)
    cnt=cnt+1

curstat=stat(chosn,[0,0,0,0,0,0,0],0)
totnum=7
eventlist=[(chosn.copy(),0)]

for t in range(1,186):
    curstat.t+=1
    curstat.ccnt=[i+1 for i in curstat.ccnt]
    temp = curstat.lorder.copy()
    lcnt = getCompltTime(temp, sdict)
    temp=curstat.ccnt
    #print(t,curstat.lorder,curstat.ccnt,lcnt)
    eventlist.append((curstat.lorder.copy(),t))
    if unequal(temp,lcnt) and ('J0' not in curstat.lorder):
        pass
    else:
        if temp[0]==lcnt[0] or curstat.lorder[0]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 0, t, stdict,
                eddict, sdict, numdict)
            if newstat[0]!='J0':
                removal(avalistsuccess,newstat[0])
                if '-1' not in newstat[0]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[0] = 0
        if temp[1]==lcnt[1] or curstat.lorder[1]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 1, t, stdict,
                eddict, sdict, numdict)
```

```
            if newstat[1]!='J0':
                removal(avalistsuccess,newstat[1])
                if '-1' not in newstat[1]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[1] = 0
        if temp[2]==lcnt[2] or curstat.lorder[2]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 2, t, stdict,
                eddict, sdict, numdict)
            if newstat[2]!='J0':
                removal(avalistsuccess,newstat[2])
                if '-1' not in newstat[2]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[2] = 0
        if temp[3]==lcnt[3] or curstat.lorder[3]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 3, t, stdict,
                eddict, sdict, numdict)
            if newstat[3]!='J0':
                removal(avalistsuccess,newstat[3])
                if '-1' not in newstat[3]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[3] = 0
        if temp[4]==lcnt[4] or curstat.lorder[4]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 4, t, stdict,
                eddict, sdict, numdict)
            if newstat[4]!='J0':
                removal(avalistsuccess,newstat[4])
                if '-1' not in newstat[4]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[4] = 0
        if temp[5]==lcnt[5] or curstat.lorder[5]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 5, t, stdict,
                eddict, sdict, numdict)
            if newstat[5]!='J0':
                removal(avalistsuccess,newstat[5])
                if '-1' not in newstat[5]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[5] = 0
        if temp[6]==lcnt[6] or curstat.lorder[6]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 6, t, stdict,
                eddict, sdict, numdict)
            if newstat[6]!='J0':
                removal(avalistsuccess,newstat[6])
                if '-1' not in newstat[6]: totnum += 1
            curstat.lorder = newstat
            curstat.ccnt[6] = 0
    #print(avalistsuccess)
print(totnum)
```

```
    maxtot=max(totnum,maxtot)
    if maxtot == totnum: finlist = eventlist
print('max',maxtot)
print('min',mintot)
print('max cost',maxcost)
print('min cost',mincost)
print('min target:',target)
print(finlist)
```

附录 B　问题二的改进状态转移模型的 Python 代码

```
import pandas as pd
import numpy as np
import enum
import random
import math

def getStageTime(frame):
    dictlist={'1J0':-1,'2J0':-1,'3J0':-1,'4J0':-1,'5J0':-1,'6J0':-1,'7J0':-1}
    for item in frame.itertuples():
        index=str(item[3])+str(item[8])
        stages=math.ceil(float(item[11])/item[4])
        dict1={index:stages}
        dict2={index+'-1':2}
        dictlist.update(dict1)
        dictlist.update(dict2)
    return dictlist

def getNumDict(frame):
    numdict={}
    numbasdict={}
    for item in frame.itertuples():
        index=item[1]
        num=item[2]
        bas=item[7]
        numdict.update({index: num})
        numbasdict.update({index: bas})
        numdict.update({index + '-1': num})
        numbasdict.update({index + '-1': bas})
    return numdict,numbasdict

def getBasCost(frame):
    basdict={}
    for item in frame.itertuples():
        index=item[1]
```

```
        basdict.update({index+frame.columns[1]:item[2]})
        basdict.update({index + frame.columns[2]: item[3]})
        basdict.update({index + frame.columns[3]: item[4]})
        basdict.update({index + frame.columns[4]: item[5]})
        basdict.update({index + frame.columns[5]: item[6]})
        basdict.update({index + frame.columns[6]: item[7]})
        basdict.update({index + frame.columns[7]: item[8]})
        basdict.update({index + frame.columns[8]: item[9]})
        basdict.update({index + frame.columns[9]: item[10]})
        basdict.update({index + frame.columns[10]: item[11]})
        basdict.update({index + frame.columns[11]: item[12]})
        basdict.update({index + frame.columns[12]: item[13]})
        basdict.update({index + frame.columns[13]: item[14]})
        basdict.update({index + frame.columns[14]: item[15]})
        basdict.update({index + frame.columns[15]: item[16]})
        basdict.update({index + frame.columns[16]: item[17]})
        basdict.update({index + frame.columns[17]: item[18]})
    return basdict

def getLimitTime(frame):
    lstdict={}
    leddict={}
    for item in frame.itertuples():
        index=item[1]
        slim=(item[3].day-pd.Timestamp('2021-08-01 00:00:00').day)*24
        elim=(item[6].day-pd.Timestamp('2021-08-01 00:00:00').day)*24-7
        lstdict.update({index:slim})
        lstdict.update({index+'-1':slim})
        leddict.update({index:elim})
        leddict.update({index+'-1':elim})
    return lstdict,leddict

def getCompltTime(lis,sdict):
    lcnt = []
    for i in range(1,8):
        lcnt.append(sdict[str(i) + lis[i-1]])
    return lcnt

def unequal(lis1,lis2):
    for i in range(len(lis1)):
        if lis1[i]==lis2[i]:
            return False
    return True

def generateChosen(curlist,avalist,index,t,stdict,eddict,sdict,numdict):
    cand=[]
    litord=['J1','J2','J3','J4','J5','J6','J7','J8','J9','J10','J11','J12','J13','J14','J15','J16',
```

```
    'J17','J18', 'J19','J20','J21','J22','J23','J24','J25','J26','J27','J28','J29']
    for item in avalist[index]:
        if curlist[index]!='J0' and numdict[item]!=numdict[curlist[index]]:
            if item in litord and stdict[item]<=t and
                eddict[item]+24>=t+sdict[str(index+1)+item]+2:
                cand.append(item)
            elif stdict[item]<=t and eddict[item]>=t+sdict[str(index+1)+item]+2:
                cand.append(item)
        elif curlist[index]!='J0' and numdict[item]==numdict[curlist[index]]:
            if item in litord and stdict[item]<=t and
                eddict[item]+24>=t+sdict[str(index+1)+item]:
                cand.append(item)
            elif stdict[item]<=t and eddict[item]>=t+sdict[str(index+1)+item]:
                cand.append(item)
    #cand = [n for n in avalist[index] if (stdict[n] <= t and eddict[n] >= t + sdict[str(index
        + 1) + n])]
    #print(cand)
    if '-1' in curlist[index]:
        curlist[index]=curlist[index].replace('-1','')
    elif len(cand)==0:
        curlist[index]='J0'
    else:
        chosn=cand[random.randint(0,len(cand)-1)]
        while (chosn in curlist) or (chosn+'-1' in curlist):
            chosn=cand[random.randint(0,len(cand)-1)]
        if curlist[index]!='J0' and numdict[chosn]!=numdict[curlist[index]]:
            chosn=chosn+'-1'
        curlist[index]=chosn
    return curlist

def removal(avalist,item):
    if item in avalist[0]: avalist[0].remove(item)
    if item in avalist[1]: avalist[1].remove(item)
    if item in avalist[2]: avalist[2].remove(item)
    if item in avalist[3]: avalist[3].remove(item)
    if item in avalist[4]: avalist[4].remove(item)
    if item in avalist[5]: avalist[5].remove(item)
    if item in avalist[6]: avalist[6].remove(item)

    if '-1' in item:
        tempitem=item.replace('-1','')
        if tempitem in avalist[0]: avalist[0].remove(tempitem)
        if tempitem in avalist[1]: avalist[1].remove(tempitem)
        if tempitem in avalist[2]: avalist[2].remove(tempitem)
        if tempitem in avalist[3]: avalist[3].remove(tempitem)
        if tempitem in avalist[4]: avalist[4].remove(tempitem)
        if tempitem in avalist[5]: avalist[5].remove(tempitem)
```

```
        if tempitem in avalist[6]: avalist[6].remove(tempitem)
    return avalist

class stat:
    def __init__(self,lorder,ccnt,t):
        self.lorder=lorder
        self.ccnt=ccnt
        self.t=t

productFrame=pd.read_excel('product.xls')
producerFrame=pd.read_excel('producer.xls')
basmc=pd.read_excel('basmcost.xls')

namelist=productFrame['产品型号'].tolist()

#touseFrame=producerFrame[producerFrame['产品型号'].isin(namelist)]
touseFrame=pd.read_excel('touseprod.xls')

usedFrame=touseFrame.merge(productFrame)

basdict=getBasCost(basmc)
sdict=getStageTime(usedFrame)
stdict,eddict=getLimitTime(productFrame)
numdict,numbasdict=getNumDict(productFrame)

avalist=[['J6','J10','J16','J22','J27','J36','J38','J39','J41','J46','J52','J53','
J56','J59','J60','J62','J63','J64'],
        ['J1','J4','J7','J11','J19','J23','J26','J31','J32','J35','J38','J42','J43',
        'J44','J46','J48','J49','J52','J54','J56','J57','J58','J61','J62','J65'],
        ['J4','J7','J8','J11','J12','J14','J26','J27','J31','J40','J41','J43','J44',
        'J45','J46','J56','J58','J62'],
        ['J3','J4','J5','J7','J11','J12','J14','J19','J20','J23','J26','J28','J29',
        'J32','J35','J37','J38','J42','J43','J45','J46','J48','J52','J56','J60'],
        ['J6','J11','J13','J15','J16','J17','J18','J21','J25','J27','J28','J30','J33',
        'J34','J35','J37','J41','J46','J51','J56','J60','J62'],
        ['J9','J12','J24','J48','J51','J55'],
        ['J2','J3','J6','J11','J13','J17','J18','J19','J21','J25','J30','J33','J34',
        'J46','J47','J50','J56','J60']]

maxtot=0
mintot=100
maxcost=0
mincost=100
finlist=[]
target=10000

for ite in range(4000):
```

```
avalistsuccess=[['J10','J27','J36','J38','J39','J41','J46','J52','J53','J56','J59',
'J60','J62','J63','J64'],
        ['J11','J31','J32','J35','J38','J42','J43','J44','J46','J48','J49','J52',
        'J54','J56','J57','J58','J61','J62','J65'],
        ['J8','J11','J12','J27','J31','J40','J41','J43','J44','J45','J46','J56'
        ,'J58','J62'],
        ['J3','J4','J5','J7','J11','J12','J14','J19','J20','J23','J26','J28','J29','J32',
        'J35','J37','J38','J42','J43','J45','J46','J48','J52','J56','J60'],
        ['J11','J27','J30','J33','J34','J35','J37','J41','J46','J51','J56','J60','J62'],
        ['J9','J12','J24','J48','J51','J55'],
        ['J2','J3','J6','J11','J13','J17','J18','J19','J21','J25','J30','J33','J34','J46',
        'J47','J50','J56','J60']]

cand=[[n for n in avalistsuccess[0] if stdict[n]==0],
       [n for n in avalistsuccess[1] if stdict[n]==0],
       [n for n in avalistsuccess[2] if stdict[n]==0],
       [n for n in avalistsuccess[3] if stdict[n]==0],
       [n for n in avalistsuccess[4] if stdict[n]==0],
       [n for n in avalistsuccess[5] if stdict[n]==0],
       [n for n in avalistsuccess[6] if stdict[n]==0]]

chosn=[]
cnt=0
while cnt<7:
    item=cand[cnt][random.randint(0,len(cand[cnt])-1)]
    while item in chosn:
        item=cand[cnt][random.randint(0,len(cand[cnt])-1)]
    chosn.append(item)
    removal(avalistsuccess,item)
    cnt=cnt+1

curstat=stat(chosn,[0,0,0,0,0,0,0],0)
totnum=7
eventlist=[(chosn.copy(),0)]
totcost=0
res=0

for t in range(1,186):
    curstat.t+=1
    curstat.ccnt=[i+1 for i in curstat.ccnt]
    temp = curstat.lorder.copy()
    lcnt = getCompltTime(temp, sdict)
    temp=curstat.ccnt
    #print(t,curstat.lorder,curstat.ccnt,lcnt)
    eventlist.append((curstat.lorder.copy(),t))
    if unequal(temp,lcnt) and ('J0' not in curstat.lorder):
        pass
```

```
    else:
        if temp[0]==lcnt[0] or curstat.lorder[0]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 0, t, stdict,
                eddict, sdict, numdict)
            if newstat[0]!='J0':
                removal(avalistsuccess,newstat[0])
                if '-1' not in newstat[0]: totnum += 1
                if curstat.lorder[0]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[0]]+numbasdict[newstat[0]]]
            curstat.lorder = newstat
            curstat.ccnt[0] = 0
        if temp[1]==lcnt[1] or curstat.lorder[1]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 1, t, stdict,
                eddict, sdict, numdict)
            if newstat[1]!='J0':
                removal(avalistsuccess,newstat[1])
                if '-1' not in newstat[1]: totnum += 1
                if curstat.lorder[1]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[1]]+numbasdict[newstat[1]]]
            curstat.lorder = newstat
            curstat.ccnt[1] = 0
        if temp[2]==lcnt[2] or curstat.lorder[2]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 2, t, stdict,
                eddict, sdict, numdict)
            if newstat[2]!='J0':
                removal(avalistsuccess,newstat[2])
                if '-1' not in newstat[2]: totnum += 1
                if curstat.lorder[2]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[2]]+numbasdict[newstat[2]]]
            curstat.lorder = newstat
            curstat.ccnt[2] = 0
        if temp[3]==lcnt[3] or curstat.lorder[3]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 3, t, stdict,
                eddict, sdict, numdict)
            if newstat[3]!='J0':
                removal(avalistsuccess,newstat[3])
                if '-1' not in newstat[3]: totnum += 1
                if curstat.lorder[3]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[3]]+numbasdict[newstat[3]]]
            curstat.lorder = newstat
            curstat.ccnt[3] = 0
        if temp[4]==lcnt[4] or curstat.lorder[4]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 4, t, stdict,
                eddict, sdict, numdict)
            if newstat[4]!='J0':
                removal(avalistsuccess,newstat[4])
                if '-1' not in newstat[4]: totnum += 1
```

```
                if curstat.lorder[4]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[4]]+numbasdict[newstat[4]]]
            curstat.lorder = newstat
            curstat.ccnt[4] = 0
        if temp[5]==lcnt[5] or curstat.lorder[5]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 5, t, stdict,
                eddict, sdict, numdict)
            if newstat[5]!='J0':
                removal(avalistsuccess,newstat[5])
                if '-1' not in newstat[5]: totnum += 1
                if curstat.lorder[5]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[5]]+numbasdict[newstat[5]]]
            curstat.lorder = newstat
            curstat.ccnt[5] = 0
        if temp[6]==lcnt[6] or curstat.lorder[6]=='J0':
            newstat = generateChosen(curstat.lorder.copy(), avalistsuccess, 6, t, stdict,
                eddict, sdict, numdict)
            if newstat[6]!='J0':
                removal(avalistsuccess,newstat[6])
                if '-1' not in newstat[6]: totnum += 1
                if curstat.lorder[6]!='J0':
                    totcost+=100-basdict[numbasdict[curstat.lorder[6]]+numbasdict[newstat[6]]]
            curstat.lorder = newstat
            curstat.ccnt[6] = 0
    #print(avalistsuccess)
  print(totnum)
  res=0.5*(totcost-50)/(250-50)+0.5*(totnum-31)/(55-31)
  target=min(res,target)
  if res==target: finlist=eventlist
print('min target:',target)
print(finlist)
```

附录 C　用于数据可视化的 Python 代码

```
  import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['font.size']='14'
# Matplotlib中设置字体-黑体，解决Matplotlib中文乱码问题
plt.rcParams['axes.unicode_minus'] = False
data=pd.read_csv("q2.csv")
time=data["时间"]
n=0
```

```
num=[]
t=[]
for j in range(1,8):
    for i in range(1,len(time)):
        if data["机台"+str(j)][i]!=data["机台"+str(j)][i-1]:
            t.append(j)
            num.append(i)
for i in range(len(num)):
    if(i%2==1):
        num[i]=num[i]-1
list=[[],[],[],[],[],[],[]]
i=0
flag=0
while(i<len(num)):
    if i%2==0 and num[i]-num[i-1]!=2:
        list[t[i]-1].append(num[i])
        print(num[i])
    i=i+1
print(list)
a=["机台1","机台2","机台3","机台4","机台5","机台6","机台7"]
sum=0

for i in range(7):
    for j in range(len(list[i])):
        if j!=0:
            sum=list[i][j-1]+2
        if j==0:
            sum=0
        # print(i,sum,list[i][j])
        plt.barh(a[i],list[i][j]-sum,left=sum,facecolor='dodgerblue',edgecolor='dodgerblue')
        plt.barh(a[i],2,left=list[i][j],facecolor="white",edgecolor='dodgerblue')
    #temp=[34,23,12,8,22,120,5]
    temp=[34,23,4,24,22,120,7]
    #temp=[34,11,1,24,7,116,100]
    #print(list[i][j]+2,list[i][j]+2+temp[i])
    if i==5:
        continue
    plt.barh(a[i],temp[i],left=list[i][j]+2,facecolor='dodgerblue',edgecolor='dodgerblue')
plt.xticks([])
plt.show()
```