水下机器人的组装计划

摘要

近年来，随着生活条件的改善，自来水管道逐渐走进千家万户. 然而，自来水管道容易在长期使用中积累污垢，对供水品质有较大影响. 本文立足于自来水管道清理机器人工厂生产条件，逐步改进其材料订购和生产计划，以降低企业生产成本，并为相关企业提供参考.

对于问题一，我们以总成本最低为目标，建立了符合工厂生产特点的线性规划模型，并制定出工厂一周最优生产计划. 首先以工厂生产成本构建目标函数，并引入 0–1 变量表示并计算生产准备费. 在题中采购和组装无延迟的前提下，为了模拟工厂生产条件，提出了一系列约束条件：库存数量约束、WPCR 需求约束、工时约束等. 最后利用 GUROBI 对模型进行求解，制定工厂的 7 天生产计划，并进行了可视化结合库存数量变化情况认为工厂频繁开工节约仓储费用.

对于问题二，针对题目中提出的组件不可直接投入生产的限制，本文在问题一的模型基础上进行改进，得到了工厂所需的优化生产计划. 由于组装产品所需的组件要提前一天入库才能参与生产，为了保障工厂每天的正常生产，添加了令工厂只得使用前一天的组件库存生产新组件的新约束. 通过求解改进过的线性规划模型，得到最优 7 天生产计划，结合图表进行评估，发现工厂前期生产量较大、但后期较小.

对于问题三，为了保障生产的持续性，工厂将在 210 天内设置 7 个检修日. 据此，本文考虑如何安排生产工厂停工检修，可令总成本最小. 在上文模型的基础上，为了表示当天是否检修，而引入逻辑 0–1 变量参与线性约束. 通过求解线性规划模型，得到了局部最优解作为检修日安排的最优方案.

对于问题四，工厂外部订单数量不再已知，需要根据历史订单数据结合上文模型制定未来生产方案. 因此，建立 ARIMA(p, T, q) 模型，以预测未来 7 天订单数. 通过进行显著性水平 α 来计算正常支付的概率，将通过检验的预测数据作为需求数代入问题 2 的优化模型求解，得出未来 7 天的生产计划.

最后我们对模型进行了中肯的评价和适当的推广.

关键字： 0–1 规划　ARIMA 模型　线性规划

近年来，随着社会经济的发展和科学技术的进步，管道越来越多的运用在我们的生活当中．管道运输是最实用、最经济的运输方式，所以管道运输在生活和生产中的使用也越来越广泛[1]．其中，自来水管道已经进入千家万户．当净水厂的出厂水经过供水管网，输送给用户时，其在供水管网中将会发生复杂的生物、物理、化学反应[2]．导致污垢在自来水管道壁上积累，影响自来水的品质和用途．自来水管道清理机器人是一种体型较小，使用机械臂以辅助完成管道清理任务的自动装置．相比与传统人工清理方式，机器人清理具有及时、高效的优点，因此倍受水务公司和住户的青睐．

一、 问题重述与分析

某工厂生产的 WPCR 装置由 3 个容器艇（用 A 表示）、4 个机器臂（用 B 表示）、5 个动力系统（用 C 表示）组装而成．而 A、B、C 由以下部件组成：

- 容器艇（A）由 6 个控制器（A1）、8 个划桨（A2）和 2 个感知器（A3）组成，组装需消耗 3 个工时；
- 机器臂（B）由 2 个力臂组件（B1）和 4 个遥感器（B2）组成，组装需消耗 5 个工时；
- 动力系统（C）由 8 个蓄电池（C1）、2 个微型发电机（C2）和 12 个发电螺旋（C3）组成，组装需消耗 5 个工时．

工厂在某一天生产组件产品时，都要付出一个与生产数量无关的固定成本，称为生产准备费用．而当一天结束时仍有某部件的库存，则须付出额外的库存费用．每次生产计划的计划期为一周，提供的最终产品为 WPCR，以满足订单需要，不可轻易缺货断供．

为了最大化经济效益，帮助生产工厂做出决策，本文建立了结合 ARIMA 预测方法的线性规划模型．

1.1 问题一

题目要求生产周期开始时没有任何组件库存，周期结束后也不留下任何组件库存．在部件采购与 WPCR 组装无延迟的基础上，要求总成本最小．因此可直接以总成本为目标函数建立线性规划模型，在题目所给约束条件下求得最优解，以制定每周 7 天的生产计划．

1.2 问题二

题目要求在问题一模型的基础上，考虑组件入库延迟对模型和生产计划的影响．与问题一组件入库无延迟不同，问题二中组装产品所需的组件要提前一天入库才能参与生产．因此，本文在问题一模型约束条件中，添加了令工厂只得使用前一天的组件库存生产新组件的限制．通过求解新模型，得到记入组件入库延迟的最优 7 天的生产计划．

1.3 问题三

题目要求在问题二模型的基础上，考虑如何安排生产工厂停工检修，可令总成本最小．总共需设置 7 次停工检修，每次检修时间为 1 天，检修之后关键设备生产能力会略有上升．因此，本文在模型约束条件中，引入了表示当天 t 是否检修的 0–1 变量 μ_t，并根据题目调整关键设备产能．在上文基础上，建立了以总成本为目标函数的线性规划模型，以确定检修日安排的最佳方案．

1.4 问题四

题目要求在以上模型的基础上，考虑 WPCR 外部需求订单未知时，如何制定合理的周生产计划．在追求周总成本最小的前提下，同时保证每周和每天正常交付率处于合理区间．

二、 符号说明

表 **1**　文中符号所用说明

符号	说明
x_t^r	第 t 天，组件 r（包括 WPCR）的组装数量.
y_t^r	第 t 天，组件 r（包括 WPCR）的库存数量.
d_t	第 t 天，WPCR 的外部需求数量.
M_t^r	第 t 天，组件 r（仅包括 A、B 和 C）的生产总工时限制.
s^r	组件 r（包括 WPCR）的生产准备费用.
h^r	组件 r（包括 WPCR）的单件库存费用.
c^r	组件 r（仅包括 A、B 和 C）的单件工时消耗.

三、 工厂生产计划设计（问题一）

3.1 模型的建立

本题要求制定每周 7 天的生产计划，使总成本最小. 工厂生产成本可分为两个部分：

1. 各部件生产的生产准备费用
   只与组件相关，与开工时间无关、任一天开工的生产准备费均相等，后文记为 s^r；
2. 各部件的存贮费用
   共有 12 种产品（$R=12$)，在每个时间阶段（本题将一周七天订为整个时间跨度 T，并等分为七个时间阶段 t）结尾，如果有产品 r 库存，则需支付相应的储存费用，单件产品 1 个时段的存贮费为 h^r.

另外，由于产品 A、B、C 的加工过程需占用关键设备，消耗的时间不可忽略，记为工时 $c^r(r\in\{\text{A,B,C}\})$.

为了使总成本最小，本文以总成本 z 为目标函数，建立线性规划模型. 目标函数如下：

$$\min z=\sum_{t=1}^{T}\sum_{r=1}^{R}\left(s^r\omega_t^r+h^ry_t^r\right). \tag{1}$$

上式中，$h^ry_t^r$ 表示产品 r 库存的存贮费，其中，y_t^r 是该产品的存贮数量；$s^r\omega_t^r$ 表示部件 r 生产的生产准备费用，这里的 ω_t^r 是为了表述在 t 时段是否生产组件 r，从而确定是否要支付生产准备费，而引用的 0–1 变量

$$\omega_t^r=\begin{cases}1, & x_t^r>0\\ 0, & x_t^r=0\end{cases},\quad t=1,2,\cdots,T,r=1,2,\cdots,R. \tag{2}$$

题目中还要求生产周期开始时没有任何组件库存，周期结束后也不留下任何组件库存，且部件采购与 WPCR 组装没有延迟，故应满足以下约束条件：

- 库存数量
  组件 r 在第 t 天的库存量，应为前一日的库存量 y_{t-1}^r 与当日组装数量 x_t^r 之和，再减去当日为生产组件 r' 而消耗的数量

$$y_t^r=y_{t-1}^r+x_t^r-\eta_{r'}^rx_t^{r'},\quad t=1,2,\cdots,T,r=1,2,\cdots,R. \tag{3}$$

  上式中，$\eta_{r'}^r$ 为组装一个产品 r' 所需特定组件 r 的数量.

- WPCR 需求量
  WPCR 每日供应量应等于当日的 WPCR 外部需求数 d_t

$$y_{t-1}^{\text{WPCR}}+x_t^{\text{WPCR}}-y_t^{\text{WPCR}}=d_t,\quad t=1,2,\cdots,T \tag{4}$$

- 逻辑变量

  为了表示在 t 时段是否生产组件 r，从而确定是否要支付生产准备费，引用 0–1 变量 ω_t^r，等于 1 表示生产，反之表示不生产

$$\omega_t^r = \begin{cases} 1, & x_t^r > 0 \\ 0, & x_t^r = 0 \end{cases}, \quad t = 1, 2, \cdots, T, r = 1, 2, \cdots, R. \tag{5}$$

  这里使用组件 r 的组装数量 x_t^r 是否大于 0，来判断是否需要生产准备费.

- 生产总工时

  产品 A、B、C 的加工过程需占用关键设备，消耗的时间不可忽略，记为工时满足下式

$$\sum_{r=1}^{R} c^r x_t^r \leqslant M_t, \quad t = 1, 2, \cdots, T, \tag{6}$$

  各部件生产工时之和应小于总生产工时.

- 题中所给边界条件

  要求生产周期开始时没有任何组件库存，周期结束后也不留下任何组件库存，因此，令：

$$y_0^r = y_T^r = 0, \quad r = 1, 2, \cdots, R. \tag{7}$$

- 非负约束

  显然，组件 r（包括 WPCR）的组装数量与库存数量应为非负数

$$x_t^r, y_t^r \geqslant 0, \quad t = 1, 2, \cdots, T, r = 1, 2, \cdots, R. \tag{8}$$

由于式5不便直接用程序计算，故将式5、6合并为下式[3]

$$\begin{gathered} c^r x_t^r = M_t^r \omega_t^r, \quad t = 1, 2, \cdots, T, r = 1, 2, \cdots, R; \\ x_t^r = x_t^r \omega_t^r, \quad t = 1, 2, \cdots, T, r = 1, 2, \cdots, R; \\ \textstyle\sum_{r=1}^{R} M_t^r \leqslant M_t; \\ M_t^r \leqslant M_t^r \omega_t^r, \quad t = 1, 2, \cdots, T, r = 1, 2, \cdots, R; \\ M_t^r \geqslant 0, \quad r = 1, 2, \cdots, R. \end{gathered} \tag{9}$$

不难发现，当 $\omega_t^r = 0$ 时，x_t^r 必然为 0, 反之亦然，此次变换为编程实现提供了便利.

表 **2**　问题一的结果

日期 \ 部件（花费）	WPCR	B	A	C	B2	B1	A2	A3	A1	C1	C2	C3	生产准备费用	生产库存费用
周一	83	332	249	416	1328	664	1992	498	1494	3328	832	4992	1200	221.7
周二	0	344	0	0	1376	688	0	0	0	0	0	0	340	557.7
周三	81	0	243	404	0	0	1944	486	1458	3232	808	4848	860	285
周四	0	0	0	0	0	0	0	0	0	0	0	0	0	85
周五	48	173	144	240	692	346	1152	288	864	1920	480	2880	1200	111.5
周六	51	203	153	255	812	406	1224	306	918	2040	510	3060	1200	200
周天	0	0	0	0	0	0	0	0	0	0	0	0	0	0
总和	263	1052	789	1315	4208	2104	6312	1578	4734	10520	2630	15780	6260.9	

3.2 模型的求解

题中要求制定每周 7 天的生产计划，故有 $T=7$. 综上，建立的线性规划模型如下：

$$
\begin{array}{lll}
\min & z=\sum\limits_{t=1}^{T}\sum\limits_{r=1}^{R}\left(s^r\omega_t^r+h^r y_t^r\right) & \\
\text{s.t.} & \sum_{r=1}^{R} M_t^r \leqslant M_t; & \\
& M_t^r \leqslant M_t^r\omega_t^r, & t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& y_t^r = y_{t-1}^r + x_t^r - \eta_r^r x_t^{r'}, & t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& \omega_t^r \in \{0,1\}, & t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& x_t^r = x_t^r\omega_t^r, & t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& c^r x_t^r = M_t^r\omega_t^r, & t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& M_t^r \geqslant 0, & r=1,2,\cdots,R; \\
& y_0^r = y_T^r = 0, & r=1,2,\cdots,R; \\
& x_t^r, y_t^r \geqslant 0, & t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& y_{t-1}^{\text{WPCR}} + x_t^{\text{WPCR}} - y_t^{\text{WPCR}} = d_t, t=1,2,\cdots,T; &
\end{array} . \tag{10}
$$

从一方面来说，本模型同时包含连续变量和整数变量，是混合整数规划. 从另一方面，由于目标函数和约束条件对于决策变量而言都是线性的，所以本模型为线性规划. 因此，本模型为混合线性规划模型，存在最优解且已有许多可靠的求解器以供使用.

3.3 生产方案展示

一般地，工厂为降低总成本有两种策略，分别是：开车型和仓储型. 极端地，选择开车型的工厂将每个时段均进行生产以满足且仅满足所在时段需求，来尽可能减少库存费；而选择仓储型的工厂会把生产任务集中安排，来尽可能减少生产准备费，代价是会损失因产品积压导致的库存费. 然而，由于关键设备总工时的限制，这两种策略不一定

能不折不扣地实施（如：实际计算发现，周日的需求不能仅靠当日生产满足、也不存在任何一天能组装出满足一周需求的产品）. 故模型的最优解正是这两种策略达到平衡时的状态.

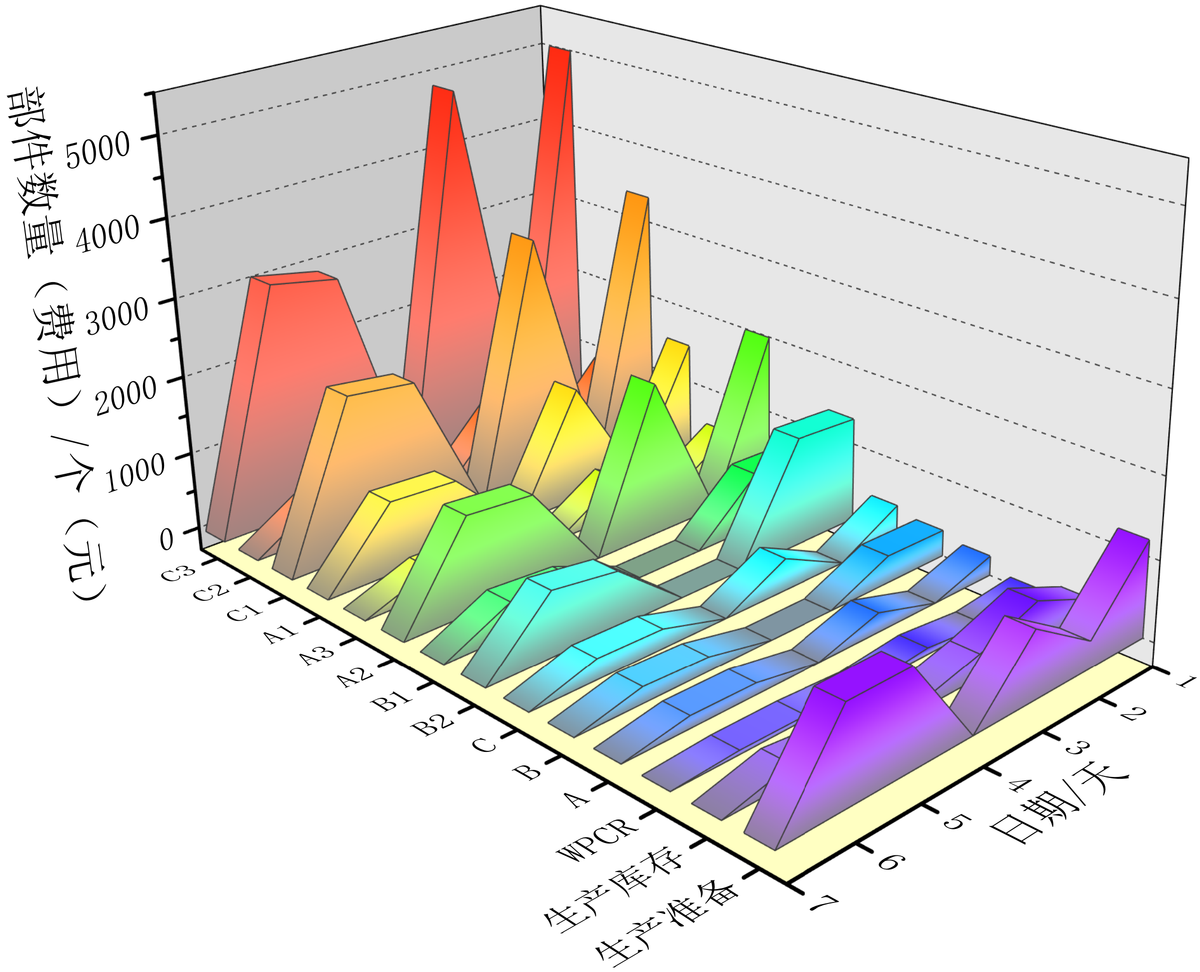

图 **1**　各组件的当日组装数量及生产费用

图 2 中未绘出的组件库存数量始终为 0，有效节省了大量库存费用；结合图 1 可知该工厂频繁开车，本文认为该工厂总体上选择了开车型策略. 这是不难理解的——其余三个问题中，库存费用远高于生产准备费、是节约成本的主要矛盾，因此工厂选用开车型策略来最小化库存费用是合理的.

然而，仓储型策略也在一定程度上参与了工厂生产的计划，周四和周日显著体现了仓储型工厂的决策特点. 周四和周日的生产总工时限制最严、但 WPCR 需求又最大，这提示生产总工时限制对决策产生的影响比 WPCR 需求大.

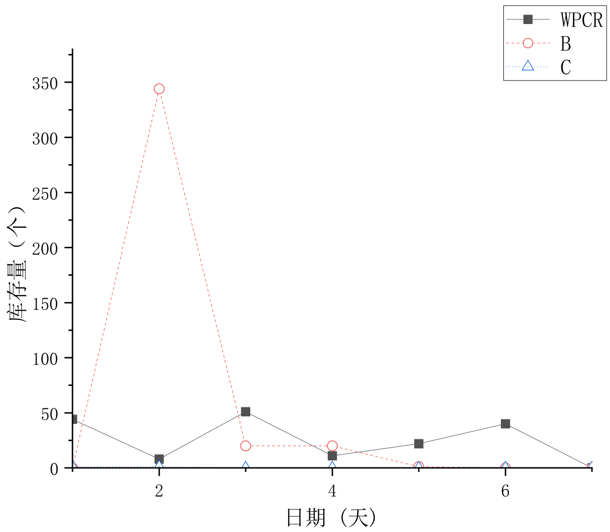

图 **2**　每日组件库存数量变化

四、 工厂生产计划扩展（问题二）

现实情况中，新采购的组件并不能直接投入使用，而是应由供应商生产、交付、转运至库存中，再供生产取用. 因此，在生产工厂发出采购订单后，订购的组件需经过一定延迟时间，才能供实际生产使用.

4.1 模型的建立与求解

本题要求在记入组件入库延迟的前提下，设计生产工厂每周 7 天的生产计划. 与问题一组件入库无延迟不同，问题二中组装产品所需的组件要提前一天入库才能参与生产.

在此基础上，由于该工厂第一天（周一）开始时没有任何组件库存，而新采购的组件在第二天（周二）才能投入生产，在第三天（周三）才可生产出 WPCR. 因此工厂必须在前一天（上周周日），准备好了刚好满足周一需求的 WPCR 库存 (即 $y_0^{\mathrm{WPCR}} = 39+36 = 75$)，以免缺货断供.

综上，本文在问题一模型基础上，添加以下约束条件：

$$\eta^{r}_{r'} x^{r'}_{t} \leqslant y^{r}_{t-1}, \quad t=1,2,\cdots,T, r=1,2,\cdots,R. \tag{11}$$

上式中，通过令工厂当天生产所需组件数量 $\eta^{r}_{r'} x^{r'}_{t}$，小于前一天库存组件 y^{r}_{t-1}，以限制工厂只得使用前一天的组件库存生产新组件.

将上式11加入问题一中线性规划模型式10，得：

$$
\begin{aligned}
\min \quad & z=\sum_{t=1}^{T}\sum_{r=1}^{R}\left(s^r\omega_t^r+h^r y_t^r\right) \\
\text{s.t.} \quad & \textstyle\sum_{r=1}^{R} M_t^r \leqslant M_t; \\
& M_t^r \leqslant M_t^r\omega_t^r, && t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& y_t^r = y_{t-1}^r + x_t^r - \eta_r^r x_t^{r'}, && t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& \omega_t^r \in \{0,1\}, && t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& x_t^r = x_t^r\omega_t^r, && t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& c^r x_t^r = M_t^r\omega_t^r, && t=1,2,\cdots,T, r=1,2,\cdots,R; \quad . \\
& M_t^r \geqslant 0, && r=1,2,\cdots,R; \\
& y_0^r = y_T^r = 0, && r=1,2,\cdots,R; \\
& x_t^r, y_t^r \geqslant 0, && t=1,2,\cdots,T, r=1,2,\cdots,R; \\
& y_{t-1}^{\text{WPCR}} + x_t^{\text{WPCR}} - y_t^{\text{WPCR}} = d_t, && t=1,2,\cdots,T; \\
& \eta_{r'}^r x_t^{r'} \leqslant y_{t-1}^r, && t=1,2,\cdots,T, r=1,2,\cdots,R;
\end{aligned}
\tag{12}
$$

在式12中，$t=1,2,\cdots,210$. 特别地，当 $t=1,2,\cdots,5$ 时，定义 $\mu_{-4}=\mu_{-3}=\mu_{-2}=\mu_{-1}=\mu_0=0$. 在安排检修时，如果希望问题 2 的生产计划保持不变，可以看出：根据问题 2 的结果，有多天无生产计划，因此可以安排检修；究竟将检修安排在哪天呢？实际上，由于检修后生产能力发生变化，把检修安排在本身没有生产任务的时段不一定是最佳的. 从题目给的条件来看，将检修安排在任何一天都是可以的. 因此模型引入了 0—1 变量，但以上模型由于规模较大，求解很慢. 所以本文并未得到全局最优解，而是在有限时间内寻找了局部最优解. 直观来看，周一的生产总工时限制宽、周六检修难以满足外部 WPCR 需求，所以检修不应安排在这两天. 为了降低库存成本，可以在其余时间中选择 WPCR 较少的七天作为起点，再通过启发式优化算法求解以上问题，这是本题的另一种解题思路. 但本文认为与线性规划相比，其缺乏确定的全局最优解，因此本文最终未选用该方法. 2

4.2 生产方案展示

问题 2 要求必须使用前一天的库存进行当日的生产工作，此时：开车型将每天生产当日所需的 WPCR、次日所需的关键组件和再次一天所需的其余组件，其中后两项即为库存；仓储型则会将集中生产 WPCR、在前一天生产关键组件并在再前一天生产所需的其余组件，形成共三天的生产计划. 但这两种计划的库存费用相等、则仓储型由于节约了大量生产准备费而占据优势，问题 2 的条件明显对仓储型有利.

由图 3 可知，工厂前期生产较多、后期较少，这是仓储型的特点. 由图 4 可知，WPCR 库存量始终几乎为 0，体现了开车型的特点. 和问题 1 不同，在问题 2 中工厂对仓储型

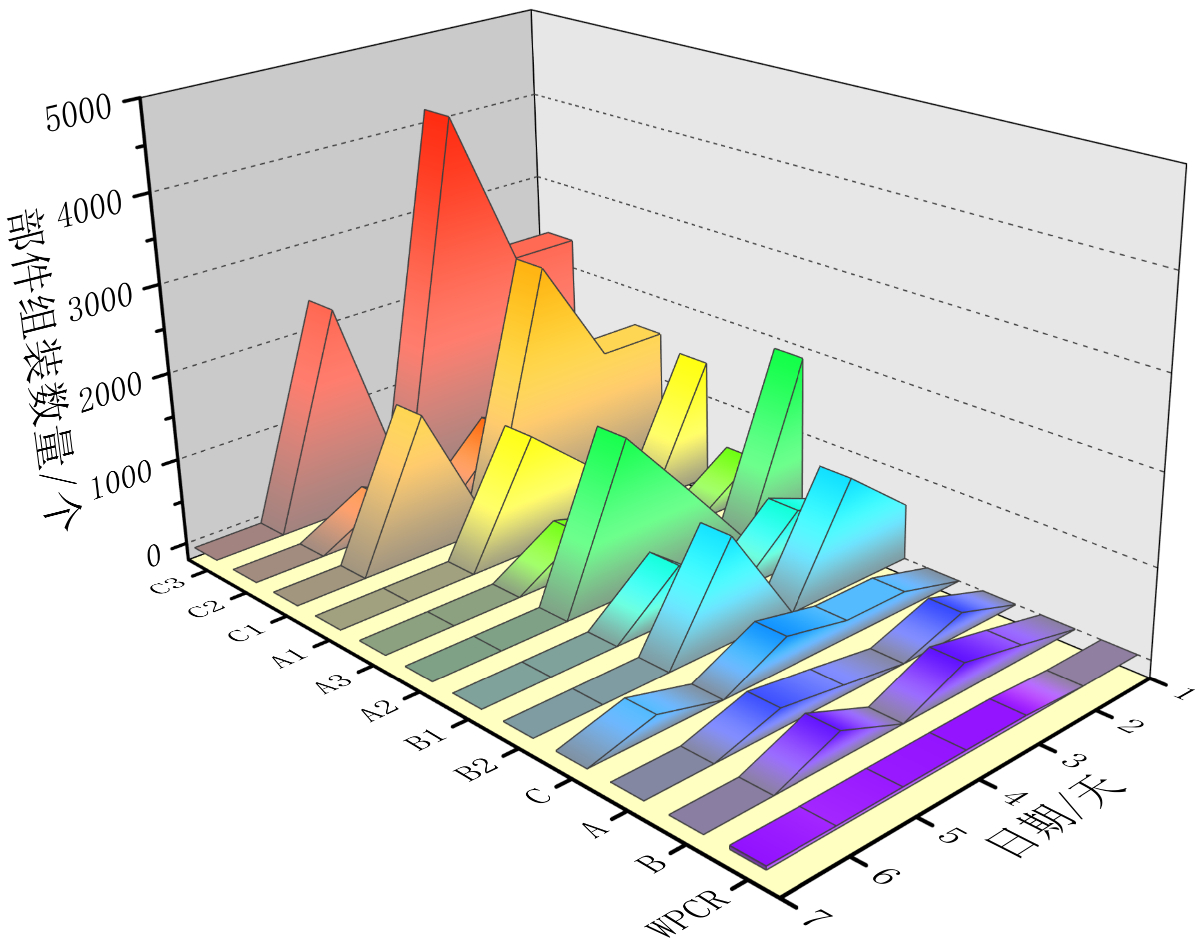

图 3　每日组件组装数量

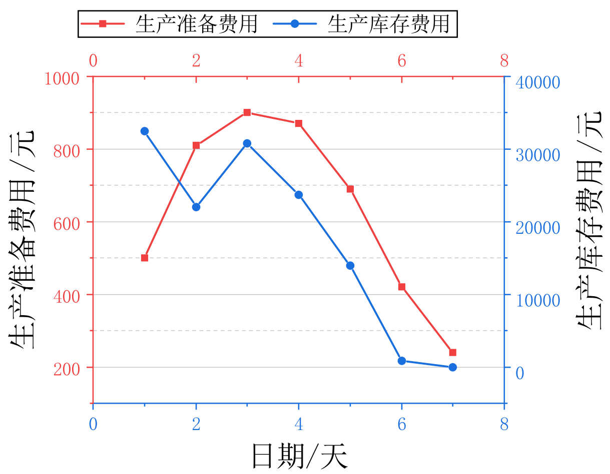

图 4　每日成本费用

表 **3** 问题二的结果

部件（花费）/日期	WPCR	B	A	C	B2	B1	A2	A3	A1	C1	C2	C3	生产准备费用	生产库存费用
周一	0	0	0	0	624	312	1872	468	1404	1560	390	2340	500	32472
周二	0	156	234	195	1216	608	0	0	0	1560	390	2340	810	22025.5
周三	39	304	0	195	0	0	888	222	666	2800	700	4200	900	30752.5
周四	39	0	111	350	1168	584	1752	438	1314	0	0	0	870	23669
周五	37	292	219	0	0	0	-2.84E-14	-7.11E-15	0	1600	400	2400	690	13956.5
周六	33	0	0	200	0	0	0	0	0	0	0	0	420	820
周天	40	0	0	0	0	0	0	0	0	0	0	0	240	0
总和	188	752	564	940	3008	1504	4512	1128	3384	7520	1880	11280	128125.5	

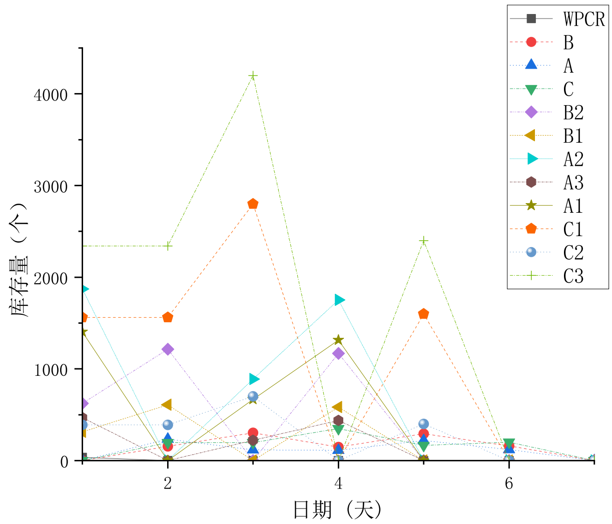

图 **5** 每日库存数量

和开车型策略都未表现出明显的倾向性，可以说问题 2 的额外要求和昂贵的仓储成本都是工厂生产计划的制订需要考虑的关键因素.

五、 检修方案设计（问题三）

为了保障生产安全、提高工厂生产条件，生产工厂常常有计划地对关键设备进行检修升级. 对关键设备的检修势必会影响其正常生产，因此有必要针对工厂具体生产条件，合理设计检修安排，保证工厂安全、高效运转.

5.1 模型的建立与求解

题目要求制定生产工厂停工检修计划，并使总成本最小. 为了表示当天 t 是否检修，本文引入了 0–1 变量 μ_t，等于 1 表示当天进行检修，等于 0 表示不进行检修

$$\mu_t \in \{0,1\}, \quad t=1,2,\cdots,210. \tag{13}$$

总共设置 7 次停工检修，每次检修时间为 1 天，故有：

$$\sum_{t=1}^{210} \mu_t = 7. \tag{14}$$

生产工厂在检修日无法生产，且两次检修需彼此间隔 6 天及以上.

$$\sum_{t=i-5}^{i} \mu_t \leqslant 1, \quad i=6,7,\cdots,210. \tag{15}$$

上式中，通过对任意 6 天内检修与否逻辑变量 μ_t 进行求和，以检验检修日间隔长度，保证检修日间隔符合题目要求.

检修之后，工厂关键设备生产效率会略有上升（工时限制放宽 10%），并以 2%/天的速率衰减到 0. 因此，需对问题一中式9总工时限制：

$$\sum_{r=1}^{R} M_t^r \leqslant M_t. \tag{16}$$

进行修正，得：

$$\sum_{r=1}^{R} M_t^r \leqslant M_t\left(1-\mu_t+0.1\mu_{t-1}+0.08\mu_{t-2}+0.06\mu_{t-3}+0.04\mu_{t-4}+0.0\mu_{t-5}\right). \tag{17}$$

其中，M_t^r 表示第 t 天、组件 r 的生产工时花费，M_t 表示计划中第 t 天生产总工时限制.

另外，本题不将一周视为一个生产周期，而将 30 周（共 210 天）视为整体，故 $T=210$.

综上，得到线性模型如下：

$$
\begin{array}{lll}
\min & z=\sum_{t=1}^{T}\sum_{r=1}^{R}(s^r\omega_t^r+h^ry_t^r) & \\
\text{s.t.} & \sum_{r=1}^{R}M_t^r\leqslant M_t(1-\mu_t+0.1\mu_{t-1}+0.08\mu_{t-2}+0.06\mu_{t-3}+0.04\mu_{t-4}+0.0\mu_{t-5}); & \\
& M_t^r\leqslant M_t^r\omega_t^r, & t=1,2,\cdots,T,r=1,2,\cdots,R; \\
& y_t^r=y_{t-1}^r+x_t^r-\eta_r^rx_t^{r'}, & t=1,2,\cdots,T,r=1,2,\cdots,R; \\
& \mu_t\in\{0,1\}; & \\
& \sum_{t=1}^{210}\mu_t=7; & \\
& \sum_{t=i-5}^{i}\mu_t\leqslant 1, & i=6,7,\cdots,210; \\
& \omega_t^r\in\{0,1\}, & t=1,2,\cdots,T,r=1,2,\cdots,R; \\
& x_t^r=x_t^r\omega_t^r, & t=1,2,\cdots,T,r=1,2,\cdots,R; \\
& c^rx_t^r=M_t^r\omega_t^r, & t=1,2,\cdots,T,r=1,2,\cdots,R; \\
& M_t^r\geqslant 0, & r=1,2,\cdots,R; \\
& x_t^r,y_t^r\geqslant 0, & t=1,2,\cdots,T,r=1,2,\cdots,R; \\
& y_{t-1}^{\text{WPCR}}+x_t^{\text{WPCR}}-y_t^{\text{WPCR}}=d_t, & t=1,2,\cdots,T; \\
& \eta_{r'}^rx_t^{r'}\leqslant y_{t-1}^r, & t=1,2,\cdots,T,r=1,2,\cdots,R;
\end{array}
\quad . \tag{18}
$$

5.2 检修方案展示

图 6 反映了产品和关键组件的组装情况变化，可看作卷曲的条形柱状图. 条柱的高度与其颜色相对应，表示每日产量占该组件 210 天内最大当日组装数量的百分比.

表 **4**　问题 **3** 的结果（检修日期及总成本）

检修日安排							总成本
第 1 次	第 2 次	第 3 次	第 4 次	第 5 次	第 6 次	第 7 次	5.3×10^6
1	37	44	51	65	86	210	

事实上，由于本文并未求得问题 3 的全局最优解，因此分析得到的局部最优解的意义是有限的. 因为边界条件约束了周一不能进行生产工作，所以将第一次检修安排在周一几乎是必然的. 根据前两问的经验，最后一天一般也不进行生产工作，故安排最后一次检修也不会损失产量，但也浪费了检修带来的产能提升. 由表四可知，其余检修安排

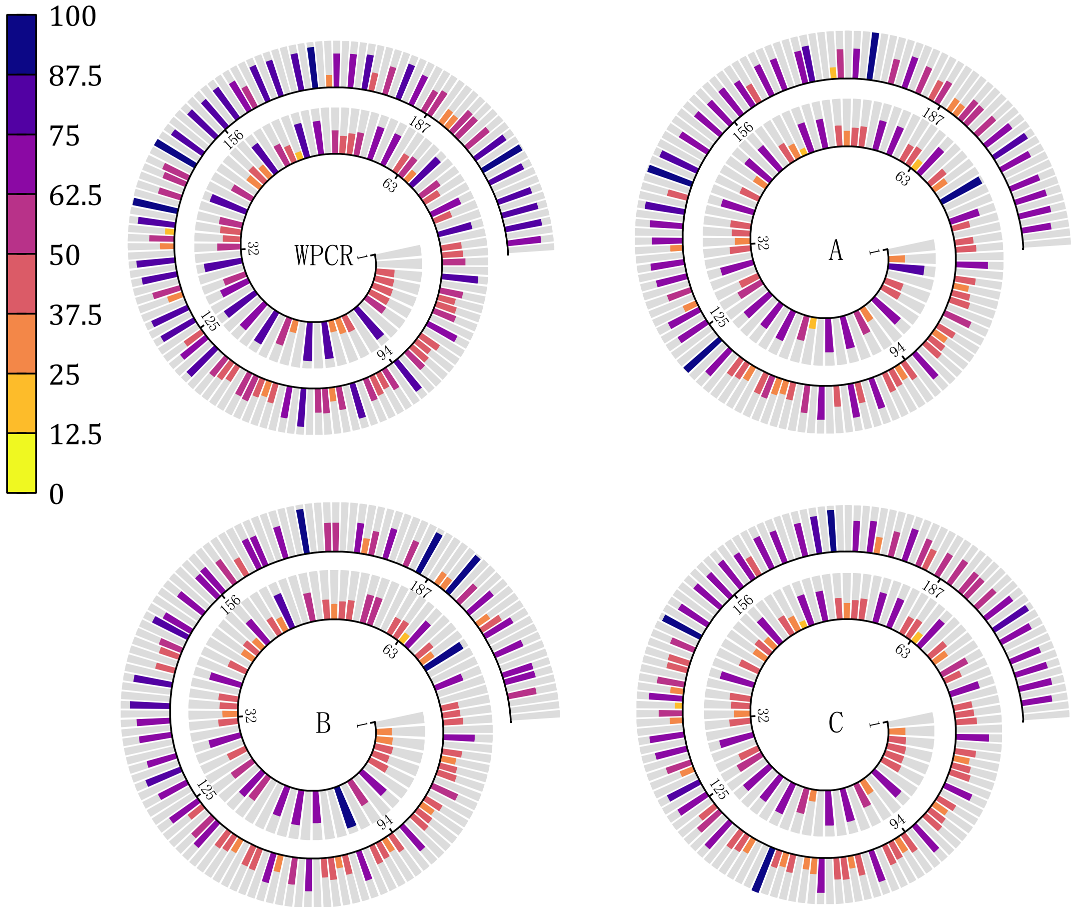

图 **6**　当日组装情况变化

在了生产中前段，且十分密集，猜测与求解器的逻辑有关. 以上安排的内在深层次原因，笔者希望在之后求得最优解后继续研究，以期得到满意解释.

六、 订单自预测型生产计划制订（问题四）

6.1 模型的建立

在未知 WPCR 需求订单前提下，公司需要根据历史周订单数据预测未来某周 7 天订单数. 将每天的 WPCR 需求数抽象为时间序列，欲对其进行时间序列分析以达到预测的目的.

因此,为了选择合理的分析方式进行预测,首先要对历史周订单数据 $d_t,\quad t = 1;2;\cdots;210$ 进行平稳性检验，即 $\phi(\mathbf{B}) = 0$ 的根是否全在单位圆外. 其中，算子 **B** 定义如下：

$$\begin{aligned} \boldsymbol{B} d_t &\equiv d_{t-1}; \\ \boldsymbol{B}^k d_t &\equiv d_{t-k}. \end{aligned} \tag{19}$$

算子多项式：

$$\phi(\boldsymbol{B})=1-\phi_1\boldsymbol{B}-\phi_2\boldsymbol{B}^2-\cdots-\phi_p\boldsymbol{B}^p,\quad p\text{ 为自回归序列的阶数}. \tag{20}$$

单位根检验有诸多方法，其中较为经典是 ADF 检验，其全称是 Augmented Dickey-Fuller test，是 Dickey-Fuller（DF）检验的扩展．DF 检验只能应用于一阶 AR 模型的情况，当序列为高阶时，存在滞后相关性，于是可以使用更适用的 ADF 检验．

事实上，在自然界中绝大部分序列都是非平稳的，此处原数据也未通过 t—检验，是非平稳时间序列．好在 Cramer 分解定理在理论上保证了适当阶数的差分一定可以充分提取确定性信息．考虑到序列蕴含着固定周期，尝试进行步长为一周的差分以提取周期信息[4]．

$$\nabla^T d_t=(1-\boldsymbol{B})^T d_t=\sum_{i=0}^{T}(-1)^i\mathbf{C}_T^i d_{t-1} \tag{21}$$

差分运算后得到的平稳序列可用 ARMA 模型进行拟合．

具有如下结构的模型称为 ARIMA(p,T,q) 模型：

$$\begin{cases}\phi(\boldsymbol{B})\nabla^T d_t=\theta(\boldsymbol{B})\varepsilon_t;\\ E\left(\varepsilon_t\right)=0,\operatorname{Var}\left(\varepsilon_t\right)=\sigma_\varepsilon^2,E\left(\varepsilon_t\varepsilon_s\right)=0,s\neq t;\\ E\left(d_t\varepsilon_t\right)=0,\forall s<t.\end{cases} \tag{22}$$

为了模型定阶（即计算 p 和 q 的值），根据 AIC 准则（又称 Akaike 信息准则，由日本统计学家 Akaike 于 1974 年提出，是信息论与统计学的重要研究成果，具有重大意义[5]）：选 p 和 q，使得

$$\min\mathrm{AIC}=n\ln\hat{\sigma}_\varepsilon^2+2(p+q+1). \tag{23}$$

其中，n 是样本容量；$\hat{\sigma}_\varepsilon^2$ 是 σ_ε^2 的估计与 p 和 q 有关. 若当 $p=\hat{p}$□$q=\hat{q}$ 时，式23达到最小值，则认为序列是 ARMA$(\hat{p},\hat{q})$.

确定好阶数后，可用矩估计、最小二乘估计或最大似然估计等方法来进行参数估计．本文不给出各种估计的数学原理和参数估计表达式，直接使用 Python 库给出相关的参数估计．对序列的预报是通过递推预报公式计算的，将理论模型中的未知参数用估计值代替，即可进行预报[6]．

为证明以上模型是否可靠、并计算正常交付的概率，最终要对模型进行 χ^2 检验. 若拟合模型的残差记为 ε_t，记

$$\rho_k=\frac{\sum_{t=1}^{n-k}\varepsilon_t\varepsilon_{t+k}}{\sum_{t=1}^{n}\varepsilon_t^2},\quad k=1,2,\cdots,L. \tag{24}$$

其中 L 为 ε_t 自相关函数的拖尾数，Ljung—Box 的 χ^2 检验统计量是

$$\chi^2=n(n+2)\sum_{k=1}^{L}\frac{\rho_k^2}{n-k} \tag{25}$$

检验的假设是：$H_0 : \rho_k = 0$, 当$k \leqslant L$ 时; $H_1 : \rho_k \neq 0$, 当某些$k \leqslant L$. 本文通过显著性水平 α 来计算正常支付的概率．并将通过检验、认为可以正常交付的预测数据作为需求数代入问题 2 的优化模型求解．

6.2 模型的求解

表 **5**　问题 **4** 结果数据

部件（花费）/日期	WPCR	B	A	C	B2	B1	A2	A3	A1	C1	C2	C3	生产准备费用	生产库存费用
周一	82	337	252	410	1348	674	2016	504	1512	3280	820	4920	1200	220.5
周二	2	0	0	485	0	0	0	0	0	3880	970	5820	590	809
周三	95	379	285	0	1516	758	2280	570	1710	0	0	0	850	255
周四	0	292	0	0	1168	584	0	0	0	0	0	0	340	453
周五	73	0	219	366	0	0	1752	438	1314	2928	732	4392	860	171.7
周六	51	204	153	254	816	408	1224	306	918	2032	508	3048	1200	225
周天	0	0	0	0	0	0	0	0	0	0	0	0	0	0
总和	303	1212	909	1515	4848	2424	7272	1818	5454	12120	3030	18180	7174.2	

经过参数估计，原数据满足的递推关系已经得到，本文根据递推预报公式得到了第 14 天到第 209 天的预测值. 并与实际观测值一同绘入图 7，在对应点间连线. 由图 7 可知，拟合趋势良好，数据间残差较小，满足预测需求，可以用于预报需求量.

由图可知，工厂产量明显较大，足以保证正常交货. 但当需求量较少时，也会导致极大地浪费，这是由于未将概率计入目标函数导致的，可以再进一步改进.

七、 模型的评价与推广

7.1 灵敏度分析

灵敏度分析是指对系统或事物因周围条件变化显示出来的敏感程度的分析．在前文讨论的线性规划模型中，都假定问题中的数据为常数．但实际上，这些参数往往是一些估计或预测的数字，经常有少许的变动．因此就会产生以下问题：

1. 当这些参数中的一个或几个发生变化时，问题的最优解会有什么变化？
2. 这些参数的变化限制在什么范围内，问题的原最优解不变.

但本文在实际应用中，是使用计算机求解的，有数值方法：给定参变量一个步长，使其重复求解线性规划问题，以观察最优解的变化情况．结果表明模型对库存费的灵敏度最高、生产准备费次之（其中 WPCR 的生产准备费又在其中最大）．但模型对其他参

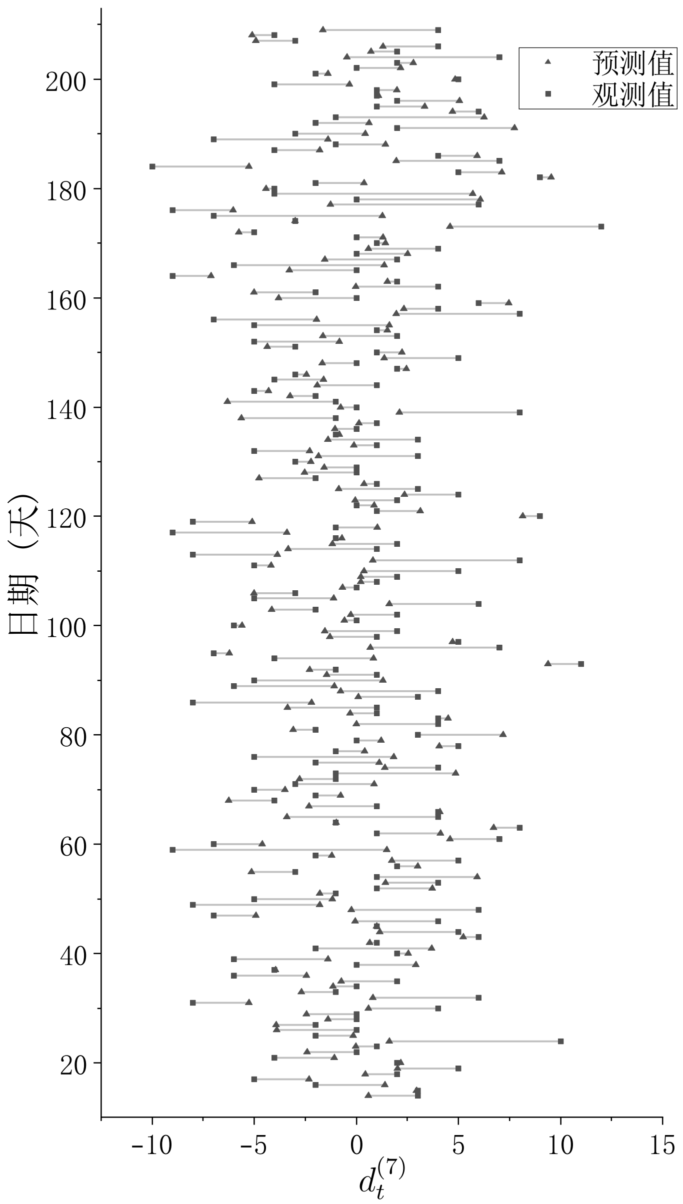

图 **7**　时序预测效果

变量并不敏感．总体来看，本文认为本模型可帮助工厂根据情况选择合适的生产计划制定策略．

7.2 模型的创新点

1. 本文通过对约束条件进行变换，维持了优化模型的比例性和确定性，从而使计算机方便处理，保留作为线性规划具有确定全局最优解的良好性质．
2. 引入了 0–1 变量解决停机检修问题，容易理解又便于计算．

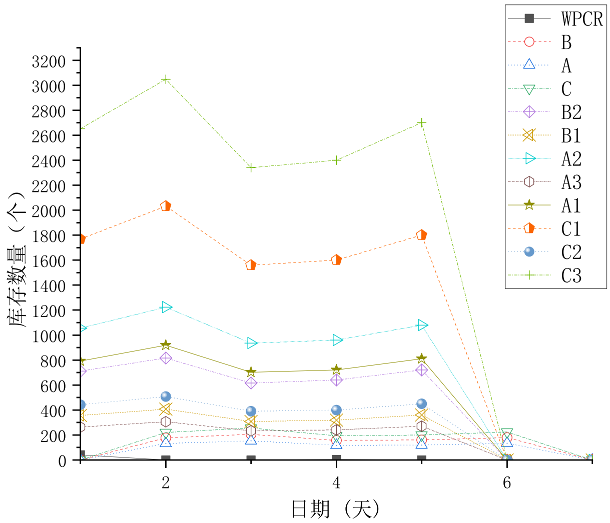

图 **8**　预测每日库存

7.3 值得改进的部分

1. 问题 3 中计算过于复杂，仅得到了局部最优解，应进一步简化算法的时间复杂度，提高优化效率.
2. 问题 4 仅以预测概率代替正常交付概率，使预测和优化分离．欲进一步考虑博弈概率模型进行多目标规划.

参考文献

[1] 刘进芬; 王志娜; 赵余婷; 刘斌; 智能管道清理机器人[J]. 物联网技术, 2021, 11(11): 96-97+100. DOI: 10.16667/j.issn.2095-1302.2021.11.028.

[2] 朱多彪; 李龙; 沈云; 基于升力法的贯流式水轮机叶片设计及可行性分析[J]. 水电能源科学, 2013, 31(07): 158-161.

[3] 姜启源, 谢金星, 叶俊, 等. 数学模型[M]. [出版地不详]: Gao deng jiao yu chu ban she, 2011.

[4] 司守奎, 孙玺菁. Python 数学实验与建模[M]. 北京: 科学出版社, 2020.

[5] 张波, 张景肖. 应用随机过程[M]. 北京: 清华大学出版社有限公司, 2004: 17-20.

[6] 胡运权. 运筹学教程[M]. 北京: 清华大学出版社有限公司, 2003: 408-422.

附录 A　源程序

1.1　水下机器人的组装计划.jl

```
using Markdown
using InteractiveUtils

using JSON, JuMP, DataFrames, CSV, Gurobi

function read_data(path::AbstractString)
    return CSV.read(path, DataFrame; transpose=true)
end

begin
    生产准备费用和单件库存费用() = read_data("每次生产准备费用和单件库存费用.csv")
    WPCR需求和关键设备工时限制() = read_data("每天 WPCR 需求和关键设备工时限制.csv")
    连续30周的WPCR需求数据() = read_data("连续 30 周的 WPCR 需求数据.csv")
end

WPCR需求和关键设备工时限制()

连续30周的WPCR需求数据()

struct Component
    生产准备费用::Float64
    单件库存费用::Float64
    begin
        function Component(生产准备费用::Float64, 单件库存费用::Float64)
            if 生产准备费用 < 0
                throw(error("生产准备费用不能为负数！"))
            end
            if 单件库存费用 < 0
                throw(error("单件库存费用不能为负数！"))
            end
            return new(生产准备费用, 单件库存费用)
        end

        function Component(生产准备费用::Number, 单件库存费用::Number)
            return Component(Float64(生产准备费用), Float64(单件库存费用))
        end
    end
end

struct Element
    生产准备费用::Float64
    单件库存费用::Float64
```

```
    工时消耗::Int64
    组件::Dict{String,Component}
    需求量::Dict{String,Int64}
    function Element(生产准备费用::Number, 单件库存费用::Number, 工时消耗::Int64,
         组件::Dict{String,Component}, 需求量::Dict{String,Int64})
        if 生产准备费用 < 0
            throw(error("生产准备费用不能为负数！"))
        end
        if 单件库存费用 < 0
            throw(error("单件库存费用不能为负数！"))
        end
        return new(Float64(生产准备费用), Float64(单件库存费用), 工时消耗, 组件, 需求量)
    end
end

struct Product
    生产准备费用::Float64
    单件库存费用::Float64
    组件::Dict{String,Element}
    需求量::Dict{String,Int64}
    function Product(生产准备费用::Number, 单件库存费用::Number, 组件::Dict{String,Element},
         需求量::Dict{String,Int64})
        if 生产准备费用 < 0
            throw(error("生产准备费用不能为负数！"))
        end
        if 单件库存费用 < 0
            throw(error("单件库存费用不能为负数！"))
        end
        return new(Float64(生产准备费用), Float64(单件库存费用), 组件, 需求量)
    end
end

struct Robot_data
    生产总工时限制::Dict{String,Int64}
    需求::Dict{String,Int64}
    产品::Product
end

function product_Element(
    df::DataFrame,
    element_name::String,
    需求量::Dict{String,Int64},
    工时消耗::Int64)

    function get_index(df::DataFrame, name)
        filter(:产品 => x -> x == name, df)[1, :]
    end
```

```
    component = Dict(let item = get_index(df, key)
        String(item[:产品]) => Component(item[:生产准备费用],
            item[:单件库存费用])
    end
                    for key in keys(需求量))
    element = let item = get_index(df, element_name)
        Element(item[:生产准备费用], item[:单件库存费用], 工时消耗, component, 需求量)
    end

    return element
end

function product_Product(
    df::DataFrame,
    product_name::String,
    需求量::Dict{String,Int64},
    组件需求量::Dict{String,Dict{String,Int64}},
    工时消耗::Dict{String,Int64})

    function get_index(df::DataFrame, name)
        filter(:产品 => x -> x == name, df)[1, :]
    end
    element = Dict(key => product_Element(df, key, 组件需求量[key], 工时消耗[key])
                for key in keys(需求量))

    product = let item = get_index(df, product_name)
        Product(item[:生产准备费用], item[:单件库存费用], element, 需求量)
    end
    return product
end

WPCR需求和关键设备工时限制()

function solve_robot(
    Optimizer,
    data,
    add_robot_variables,
    add_robot_constraints,
    add_robot_objective,
    TimeLimit=1)

    model = Model(Optimizer)
    set_optimizer_attribute(model, "TimeLimit", TimeLimit)
    add_robot_variables(model, data)
    add_robot_constraints(model, data)
    add_robot_objective(model, data)
```

```julia
    optimize!(model)
    println(termination_status(model))
    if termination_status(model) != OPTIMAL
        @warn("模型未能求解！")
        return model
    end

    return model
end

function add_robot_variables(model, data)

    time = length(data.:需求)

    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]

    let element_key = keys(element),
        component_key = keys.(components) |> x -> [[i for i in v] for v in x] |> x ->
            vcat(x...)

        @variable(model, x[["WPCR", element_key..., component_key...], [i for i in 1:time]]
            >= 0, Int)
        @variable(model, y[["WPCR", element_key..., component_key...], [i for i in 1:time]]
            >= 0, Int)
        @variable(model,  [["WPCR", element_key..., component_key...], [i for i in 1:time]],
            Bin)
        @variable(model, M[[element_key...], [i for i in 1:time]] >= 0, Int)
    end
    return model, data
end

function add_robot_constraints(model, data)
    time = length(data.:需求)

    int2week(int) = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"][int]
    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]
```

```
let element_key = keys(element), r  = "WPCR"
    for t in 1:time
        for r in element_key
            let item = element[r], r  = r, component_key = keys(item.:组件)
                for r in component_key
                    if t == 1
                        @constraint(model, model.obj_dict[:y][r, t] == 0 +
                             model.obj_dict[:x][r, t] - item.:需求量[r] *
                             model.obj_dict[:x][r , t])
                    elseif t == 7
                        @constraint(model, model.obj_dict[:y][r, t] ==
                             model.obj_dict[:y][r, t-1] + model.obj_dict[:x][r, t] -
                             item.:需求量[r] * model.obj_dict[:x][r , t])

                        @constraint(model, model.obj_dict[:y][r, t] == 0)
                    else
                        @constraint(model, model.obj_dict[:y][r, t] ==
                             model.obj_dict[:y][r, t-1] + model.obj_dict[:x][r, t] -
                             item.:需求量[r] * model.obj_dict[:x][r , t])
                    end

                    @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r,
                         t] * model.obj_dict[: ][r, t])
                end

            end

            if t == 1
                @constraint(model, model.obj_dict[:y][r, t] == 0 + model.obj_dict[:x][r,
                     t] - product.:需求量[r] * model.obj_dict[:x][r , t])
            elseif t == 7
                @constraint(model, model.obj_dict[:y][r, t] == model.obj_dict[:y][r, t-1]
                     + model.obj_dict[:x][r, t] - product.:需求量[r] *
                     model.obj_dict[:x][r , t])

                @constraint(model, model.obj_dict[:y][r, t] == 0)
            else
                @constraint(model, model.obj_dict[:y][r, t] == model.obj_dict[:y][r, t-1]
                     + model.obj_dict[:x][r, t] - product.:需求量[r] *
                     model.obj_dict[:x][r , t])
            end

            @constraint(model, model.obj_dict[:x][r, t] * element[r].:工时消耗 ==
                 model.obj_dict[:M][r, t] * model.obj_dict[: ][r, t])

            @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r, t] *
```

```
                model.obj_dict[: ][r, t])
        end

        let r = "WPCR"
            if t == 1
                @constraint(model, data.:需求[int2week(t)] == 0 + model.obj_dict[:x][r, t]
                    - model.obj_dict[:y][r, t])
            elseif t == 7
                @constraint(model, data.:需求[int2week(t)] == model.obj_dict[:y][r, t-1] +
                    model.obj_dict[:x][r, t] - model.obj_dict[:y][r, t])

                @constraint(model, model.obj_dict[:y][r, t] == 0)
            else
                @constraint(model, data.:需求[int2week(t)] == model.obj_dict[:y][r, t-1] +
                    model.obj_dict[:x][r, t] - model.obj_dict[:y][r, t])
            end

            @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r, t] *
                model.obj_dict[: ][r, t])
        end

        @constraint(model, sum(model.obj_dict[:M][r, t] for r in element_key) <=
            data.:生产总工时限制[int2week(t)])
      end
    end
    return model, data
end

function add_robot_objective(model, data)
    time = length(data.:需求)

    int2week(int) = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"][int]
    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]

    excp = 0
    let element_key = keys(element), r  = "WPCR"
      for t in 1:time
        for r in element_key
            let item = element[r], r  = r, component_key = keys(item.:组件)
                for r in component_key
                    excp += item.:组件[r].生产准备费用 * model.obj_dict[: ][r, t] +
                        item.:组件[r].单件库存费用 * model.obj_dict[:y][r, t]
```

```
                end
            end
            excp += product.:组件[r].生产准备费用 * model.obj_dict[:][r, t] +
                product.:组件[r].单件库存费用 * model.obj_dict[:y][r, t]
        end
        let r = "WPCR"
            excp += product.生产准备费用 * model.obj_dict[:][r, t] + product.单件库存费用
                * model.obj_dict[:y][r, t]
        end
    end
end
@objective(model, Min, excp)
return model, data
end

function get_生产准备费用(model, data, time)

    int2week(int) = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"][int]
    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]

    excp = 0
    t = time
    let element_key = keys(element), r  = "WPCR"
        for r in element_key
            let item = element[r], r  = r, component_key = keys(item.:组件)
                for r in component_key
                    excp += item.:组件[r].生产准备费用 * model.obj_dict[:][r, t]
                end
            end
            excp += product.:组件[r].生产准备费用 * model.obj_dict[:][r, t]
        end
        let r = "WPCR"
            excp += product.生产准备费用 * model.obj_dict[:][r, t]
        end
    end
    return excp
end

function get_生产库存费用(model, data, time)

    int2week(int) = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"][int]
    get_element(data) = data.:产品.:组件
```

```
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]

    excp = 0
    t = time
    let element_key = keys(element), r  = "WPCR"
        for r in element_key
            let item = element[r], r  = r, component_key = keys(item.:组件)
                for r in component_key
                    excp += item.:组件[r].单件库存费用 * model.obj_dict[:y][r, t]
                end
            end
            excp += product.:组件[r].单件库存费用 * model.obj_dict[:y][r, t]
        end
        let r = "WPCR"
            excp += product.单件库存费用 * model.obj_dict[:y][r, t]
        end
    end
    return excp
end

md"""
## 第一题
"""

function answer2df(model, data)
    answser = value.(model.obj_dict[:x])
    answser = answser |> Matrix |> transpose |> x -> DataFrame(x, axes(answser)[1])
    answser[!, "生产准备费用"] = (x -> get_生产准备费用(model, data, x)).([1, 2, 3, 4, 5, 6,
        7]) .|> value
    answser[!, "生产库存费用"] = (x -> get_生产库存费用(model, data, x)).([1, 2, 3, 4, 5, 6,
        7]) .|> value
    return answser
end

let data = let data = WPCR需求和关键设备工时限制(), (len, _) = size(data)
        let 生产总工时限制 = Dict((
                let item = data[i, :]
                    String(item[:天]) => item["A、B、C生产总工时限制（工时）"]
                end
                for i in 1:len
            )),
            WPCR需求 = Dict(let item = data[i, :]
                String(item[:天]) => item["WPCR需求（个）"]
```

```
            end
                        for i in 1:len),
            需求量 = Dict(
                "A" => 3,
                "B" => 4,
                "C" => 5),
            组件需求量 = Dict(
                "A" => Dict(
                    "A1" => 6,
                    "A2" => 8,
                    "A3" => 2),
                "B" => Dict(
                    "B1" => 2,
                    "B2" => 4),
                "C" => Dict(
                    "C1" => 8,
                    "C2" => 2,
                    "C3" => 12)),
            工时消耗 = Dict(
                "A" => 3,
                "B" => 5,
                "C" => 5),
            product = product_Product(生产准备费用和单件库存费用(),
                "WPCR",
                需求量,
                组件需求量,
                工时消耗)

            Robot_data(
                生产总工时限制,
                WPCR需求,
                product)
        end
    end,
    model = solve_robot(
        Gurobi.Optimizer,
        data,
        add_robot_variables,
        add_robot_constraints,
        add_robot_objective)

    CSV.write("第一题.csv", answer2df(model, data))
end

md"""
## 第二题
"""
```

```
function add_robot_constraints_2(model, data)
    time = length(data.:需求)

    int2week(int) = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"][int]
    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]

    let element_key = keys(element), r  = "WPCR"
        for t in 1:time
            for r in element_key
                let item = element[r], r  = r, component_key = keys(item.:组件)
                    for r in component_key
                        if t == 1
                            @constraint(model, model.obj_dict[:y][r, t] == 0 +
                                model.obj_dict[:x][r, t] - item.:需求量[r] *
                                model.obj_dict[:x][r , t])

                            @constraint(model, item.:需求量[r] * model.obj_dict[:x][r , t] <= 0)
                        elseif t == 7
                            @constraint(model, model.obj_dict[:y][r, t] ==
                                model.obj_dict[:y][r, t-1] + model.obj_dict[:x][r, t] -
                                item.:需求量[r] * model.obj_dict[:x][r , t])

                            @constraint(model, model.obj_dict[:y][r, t] == 0)

                            @constraint(model, item.:需求量[r] * model.obj_dict[:x][r , t] <=
                                model.obj_dict[:y][r, t-1])
                        else
                            @constraint(model, model.obj_dict[:y][r, t] ==
                                model.obj_dict[:y][r, t-1] + model.obj_dict[:x][r, t] -
                                item.:需求量[r] * model.obj_dict[:x][r , t])

                            @constraint(model, item.:需求量[r] * model.obj_dict[:x][r , t] <=
                                model.obj_dict[:y][r, t-1])
                        end

                        @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r,
                            t] * model.obj_dict[: ][r, t])
                    end

                end
```

```
    if t == 1
        @constraint(model, model.obj_dict[:y][r, t] == 0 + model.obj_dict[:x][r,
            t] - product.:需求量[r] * model.obj_dict[:x][r , t])

        @constraint(model, product.:需求量[r] * model.obj_dict[:x][r , t] <= 0)
    elseif t == 7
        @constraint(model, model.obj_dict[:y][r, t] == model.obj_dict[:y][r, t-1]
            + model.obj_dict[:x][r, t] - product.:需求量[r] *
            model.obj_dict[:x][r , t])

        @constraint(model, model.obj_dict[:y][r, t] == 0)

        @constraint(model, product.:需求量[r] * model.obj_dict[:x][r , t] <=
            model.obj_dict[:y][r, t-1])
    else
        @constraint(model, model.obj_dict[:y][r, t] == model.obj_dict[:y][r, t-1]
            + model.obj_dict[:x][r, t] - product.:需求量[r] *
            model.obj_dict[:x][r , t])

        @constraint(model, product.:需求量[r] * model.obj_dict[:x][r , t] <=
            model.obj_dict[:y][r, t-1])
    end

    @constraint(model, model.obj_dict[:x][r, t] * element[r].:工时消耗 ==
        model.obj_dict[:M][r, t] * model.obj_dict[: ][r, t])

    @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r, t] *
        model.obj_dict[: ][r, t])
end

let r = "WPCR"
    if t == 1
        @constraint(model, data.:需求[int2week(t)] == 75 + model.obj_dict[:x][r,
            t] - model.obj_dict[:y][r, t])
    elseif t == 7
        @constraint(model, data.:需求[int2week(t)] == model.obj_dict[:y][r, t-1] +
            model.obj_dict[:x][r, t] - model.obj_dict[:y][r, t])

        @constraint(model, model.obj_dict[:y][r, t] == 0)
    else
        @constraint(model, data.:需求[int2week(t)] == model.obj_dict[:y][r, t-1] +
            model.obj_dict[:x][r, t] - model.obj_dict[:y][r, t])
    end
    @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r, t] *
        model.obj_dict[: ][r, t])
end
```

```
            @constraint(model, sum(model.obj_dict[:M][r, t] for r in element_key) <=
                data.:生产总工时限制[int2week(t)])
        end
    end
    return model, data
end

let data = let data = WPCR需求和关键设备工时限制(), (len, _) = size(data)
        let 生产总工时限制 = Dict((
                let item = data[i, :]
                    String(item[:天]) => item["A、B、C生产总工时限制（工时）"]
                end
                for i in 1:len
            )),
            WPCR需求 = Dict(let item = data[i, :]
                String(item[:天]) => item["WPCR需求（个）"]
            end
                        for i in 1:len),
            需求量 = Dict(
                "A" => 3,
                "B" => 4,
                "C" => 5),
            组件需求量 = Dict(
                "A" => Dict(
                    "A1" => 6,
                    "A2" => 8,
                    "A3" => 2),
                "B" => Dict(
                    "B1" => 2,
                    "B2" => 4),
                "C" => Dict(
                    "C1" => 8,
                    "C2" => 2,
                    "C3" => 12)),
            工时消耗 = Dict(
                "A" => 3,
                "B" => 5,
                "C" => 5),
            product = product_Product(生产准备费用和单件库存费用(),
                "WPCR",
                需求量,
                组件需求量,
                工时消耗)

            Robot_data(
                生产总工时限制,
                WPCR需求,
```

```
            product)
        end
    end,
    model = solve_robot(
        Gurobi.Optimizer,
        data,
        add_robot_variables,
        add_robot_constraints_2,
        add_robot_objective)

    CSV.write("第二题.csv", answer2df(model, data))
end

md"""
## 第三题
"""

function third_write(model, path)
    third_answer(model) = model.obj_dict[:][:, 1] |> x -> zip(axes(x)[1],
        Vector(value.(x))) |> collect |> x -> filter(x -> x[2] == 1, x) |> x -> map(x ->
        x[1], x)

    title = append!(["第" * string(i) * "次" for i in 1:7], ["总成本\n"]) |> x -> join(x, ",
        ")
    values = append!(third_answer(model), [round(objective_value(model))]) |> x -> join(x,
        ", ")
    write(path, title * values)
end

function str2week(key_str)
    last = lastindex(key_str)
    first = prevind(key_str, last)
    return key_str[first:last]
end

function WPCR_30_to_Dict(data::DataFrame)
    WPCR_30_Dict = Dict()
    for (k1, v) in zip(1:100, eachcol(连续30周的WPCR需求数据()))
        int2week(int) = "第" * string(int - 1) * "周"
        int2week(int1, int2) = int2week(int1) * (["周一", "周二", "周三", "周四", "周五",
            "周六", "周日"][int2])
        if k1 != 1
            for (k2, v2) in zip(1:7, v)
                WPCR_30_Dict[int2week(k1, k2)] = v2
            end
        end
    end
```

```
    return WPCR_30_Dict
end

function add_robot_variables_3(model, data)

    time = length(data.:需求)

    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)
    components = [get_component(v) for (k, v) in element]

    let element_key = keys(element),
        component_key = keys.(components) |> x -> [[i for i in v] for v in x] |> x ->
            vcat(x...)

        @variable(model, x[["WPCR", element_key..., component_key...], [i for i in 1:time]]
            >= 0, Int)
        @variable(model, y[["WPCR", element_key..., component_key...], [i for i in 1:time]]
            >= 0, Int)
        @variable(model,  [["WPCR", element_key..., component_key...], [i for i in 1:time]],
            Bin)
        @variable(model, M[[element_key...], [i for i in 1:time]] >= 0, Int)
        @variable(model,  [[i for i in -4:time]], Bin)
        @constraint(model, [i = -4:0], model.obj_dict[:][i] == 0)
    end
    return model, data
end

function add_robot_constraints_3(model, data)
    time = length(data.:需求)

    week(day) = mod(day - 1, 7) + 1
    int2week(int) = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"][week(int)]
    day2week(day) = "第" * string(div(day - 1, 7) + 1) * "周" * int2week(day)
    get_element(data) = data.:产品.:组件
    get_component(element) = element.:组件

    product = data.:产品
    element = get_element(data)

    let element_key = keys(element), r  = "WPCR"
        for t in 1:time
            for r in element_key
                let item = element[r], r  = r, component_key = keys(item.:组件)
```

```
    for r in component_key
        if t == 1
            @constraint(model, model.obj_dict[:y][r, t] == 0 +
                model.obj_dict[:x][r, t] - item.:需求量[r] *
                model.obj_dict[:x][r , t])

            @constraint(model, item.:需求量[r] * model.obj_dict[:x][r , t] <= 0)
        elseif t == time
            @constraint(model, model.obj_dict[:y][r, t] ==
                model.obj_dict[:y][r, t-1] + model.obj_dict[:x][r, t] -
                item.:需求量[r] * model.obj_dict[:x][r , t])

            @constraint(model, model.obj_dict[:y][r, t] == 0)

            @constraint(model, item.:需求量[r] * model.obj_dict[:x][r , t] <=
                model.obj_dict[:y][r, t-1])
        else
            @constraint(model, model.obj_dict[:y][r, t] ==
                model.obj_dict[:y][r, t-1] + model.obj_dict[:x][r, t] -
                item.:需求量[r] * model.obj_dict[:x][r , t])

            @constraint(model, item.:需求量[r] * model.obj_dict[:x][r , t] <=
                model.obj_dict[:y][r, t-1])
        end

        @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r,
            t] * model.obj_dict[: ][r, t])
    end

end

if t == 1
    @constraint(model, model.obj_dict[:y][r, t] == 0 + model.obj_dict[:x][r,
        t] - product.:需求量[r] * model.obj_dict[:x][r , t])

    @constraint(model, product.:需求量[r] * model.obj_dict[:x][r , t] <= 0)
elseif t == time
    @constraint(model, model.obj_dict[:y][r, t] == model.obj_dict[:y][r, t-1]
        + model.obj_dict[:x][r, t] - product.:需求量[r] *
        model.obj_dict[:x][r , t])

    @constraint(model, model.obj_dict[:y][r, t] == 0)

    @constraint(model, product.:需求量[r] * model.obj_dict[:x][r , t] <=
        model.obj_dict[:y][r, t-1])
else
    @constraint(model, model.obj_dict[:y][r, t] == model.obj_dict[:y][r, t-1]
```

```
                + model.obj_dict[:x][r, t] - product.:需求量[r] *
                model.obj_dict[:x][r , t])

            @constraint(model, product.:需求量[r] * model.obj_dict[:x][r , t] <=
                model.obj_dict[:y][r, t-1])
        end

        @constraint(model, model.obj_dict[:x][r, t] * element[r].:工时消耗 ==
            model.obj_dict[:M][r, t] * model.obj_dict[: ][r, t])

        @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r, t] *
            model.obj_dict[: ][r, t])
    end

    let r = "WPCR"
        if t == 1
            @constraint(model, data.:需求[day2week(t)] == 75 + model.obj_dict[:x][r,
                t] - model.obj_dict[:y][r, t])
        elseif t == time
            @constraint(model, data.:需求[day2week(t)] == model.obj_dict[:y][r, t-1] +
                model.obj_dict[:x][r, t] - model.obj_dict[:y][r, t])

            @constraint(model, model.obj_dict[:y][r, t] == 0)
        else
            @constraint(model, data.:需求[day2week(t)] == model.obj_dict[:y][r, t-1] +
                model.obj_dict[:x][r, t] - model.obj_dict[:y][r, t])
        end

        @constraint(model, model.obj_dict[:x][r, t] == model.obj_dict[:x][r, t] *
            model.obj_dict[: ][r, t])
    end

    @constraint(model, sum(model.obj_dict[:M][r, t] for r in element_key) <=
        data.:生产总工时限制[int2week(t)] * (1 - model.obj_dict[: ][t] + 0.1 *
        model.obj_dict[: ][t-1] + 0.08 * model.obj_dict[: ][t-2] + 0.06 *
        model.obj_dict[: ][t-3] + 0.04 * model.obj_dict[: ][t-4] + 0.02 *
        model.obj_dict[: ][t-5]))

    @constraint(model, sum(model.obj_dict[:M][r, t] for r in element_key) <=
        data.:生产总工时限制[int2week(t)])

    if t >= 6
        @constraint(model, sum(model.obj_dict[: ][t] for t in t-5:t) <= 1)
    end

end
```

```
        @constraint(model, sum(model.obj_dict[:][t] for t in 1:time) == 7)

    end
    return model, data
end

let data = let data = WPCR需求和关键设备工时限制(), (len, _) = size(data)
        let 生产总工时限制 = Dict((
                let item = data[i, :]
                    String(item[:天]) => item["A、B、C生产总工时限制（工时）"]
                end
                for i in 1:len
            )),
            WPCR需求 = WPCR_30_to_Dict(连续30周的WPCR需求数据()),
            需求量 = Dict(
                "A" => 3,
                "B" => 4,
                "C" => 5),
            组件需求量 = Dict(
                "A" => Dict(
                    "A1" => 6,
                    "A2" => 8,
                    "A3" => 2),
                "B" => Dict(
                    "B1" => 2,
                    "B2" => 4),
                "C" => Dict(
                    "C1" => 8,
                    "C2" => 2,
                    "C3" => 12)),
            工时消耗 = Dict(
                "A" => 3,
                "B" => 5,
                "C" => 5),
            product = product_Product(生产准备费用和单件库存费用(),
                "WPCR",
                需求量,
                组件需求量,
                工时消耗)

            Robot_data(
                生产总工时限制,
                WPCR需求,
                product)
        end
    end
    model = solve_robot(
```

```
        Gurobi.Optimizer,
        data,
        add_robot_variables_3,
        add_robot_constraints_3,
        add_robot_objective,
        600)
    third_write(model, "第三题.csv")
end

md"""
## 第四题
"""

第四题预测数据() = read_data("第四题 WPCR 需求和关键设备工时限制.csv")

let data = let data = 第四题预测数据(), (len, _) = size(data)
        let 生产总工时限制 = Dict((
                let item = data[i, :]
                    String(item[:天]) => item["A、B、C生产总工时限制（工时）"]
                end
                for i in 1:len
            )),
            WPCR需求 = Dict(let item = data[i, :]
                String(item[:天]) => item["WPCR需求（个）"]
            end
                        for i in 1:len),
            需求量 = Dict(
                "A" => 3,
                "B" => 4,
                "C" => 5),
            组件需求量 = Dict(
                "A" => Dict(
                    "A1" => 6,
                    "A2" => 8,
                    "A3" => 2),
                "B" => Dict(
                    "B1" => 2,
                    "B2" => 4),
                "C" => Dict(
                    "C1" => 8,
                    "C2" => 2,
                    "C3" => 12)),
            工时消耗 = Dict(
                "A" => 3,
                "B" => 5,
                "C" => 5),
            product = product_Product(生产准备费用和单件库存费用(),
```

```
            "WPCR",
            需求量,
            组件需求量,
            工时消耗)

          Robot_data(
            生产总工时限制,
            WPCR需求,
            product)
        end
    end,
    model = solve_robot(
        Gurobi.Optimizer,
        data,
        add_robot_variables,
        add_robot_constraints,
        add_robot_objective)

    CSV.write("第四题.csv", answer2df(model, data))
end
```