基于模拟退火算法的 RGV 动态调度研究

摘要

本文针对不同加工情况下智能 RGV 动态调动问题，构建了基于最短路径贪心算法、及改良的模拟退火算法的动态调度模型，运用了 MATLAB 编程求解以及 FlexSim 模拟加工过程，给出针对一道工序，两道工序以及考虑故障情况时一道工序和两道工序四种 RGV 动态调度方案，以及两道工序下分配 8 台 CNC 装配不同加工刀具排布的方案，并给出以绝对理想情况为比较依据的系统的作业效率。

本文的突出特点是使用了 RGV 每次移动均考虑当前最优解的贪心算法，以及避免暂时困于局部最优解的模拟退火算法，寻找局部乃至全局最优解，并且遍历 CNC 的所有工序分配方案以优化最优解。

针对不考虑故障时一道工序的情况，我们采用贪心算法的思想建立**模型 I**—基于贪心算法求解最优动态调度方案模型，由于 CNC 加工时长远大于 RGV 移动以及上下料时间，此时得到的局部最优解即为全局最优解，经 3 组数据检验，系统的作业效率约为 92%，已经达到了全局最优。

针对不考虑故障时两道工序的情况，我们基于贪心算法的思想建立**模型 II**—基于贪心算法求解最优动态调度方案模型，遍历加工不同分组下不同工序 CNC 数量以及排布的 254 种情况，以优化局部最优解，得到对于 CNC 分类以及排布的最优解，发现不同数据组下存在并行最优解以及唯一最优解。此模型下系统的作业效率约为 89%。

针对考虑故障时一道工序的情况，首先创新性地定义了考虑故障时一道工序情况下 CNC 剩余工作时间矩阵，基于模拟退火算法建立了**模型 III**—基于模拟退火算法求解最优故障动态调度方案模型：一道工序。当给 CNC 进行上料操作时，对 CNC 进行一次故障检测，若有故障，则通过均匀分布生成此次故障发生的具体时间以及人工需要维护的时间，然后根据贪心算法重新选择最优路径，寻找局部最优解，并以一定的概率寻找倒数第二短的可行路径，多次运行以求得全局最优解。此时系统的作业效率约为 89%。

针对考虑故障时两道工序的情况，定义了考虑故障时两道工序情况下 CNC 剩余工作时间矩阵，基于模拟退火算法的思想建立了**模型 IV**—基于模拟退火算法求解最优故障动态调度方案模型：两道工序。在模型 III 中已经寻找到针对三组不同数据的最佳工序分配方案，模型 IV 延续此方案并对模型 III 的思想进行推广。经数据检验，模型较好的实现了有故障发生时两道工序情况下 RGV 的动态调度，系统的作业效率约为 85%。

本文为进一步增加可视化程度，保证逻辑与运算结构的清晰性，运用 FlexSim 进行仿真模拟，绘制模型的算法流程图和 RGV 动态调动示意图。本文逻辑清晰，创新性地定义了考虑故障时 CNC 的剩余工作时间矩阵，统一了模型 I 和模型 III、模型 II 和模型 IV 的表达式，合理进行误差评估以及灵敏度分析，拟合实际情况并加以推广。

关键字： 模拟退火算法；贪心算法；最短路径；RGV；动态调度；作业效率

§1 问题重述

1.1 背景知识

1.1.1 引言部分

智能加工系统是一种由智能机器和人类专家共同组成的人机一体化智能系统，它在制造过程中能以一种高度柔性与集成度高的方式，借助计算机模拟人类专家的智能活动进行分析、推理、判断、构思和决策等，从而取代或者延伸制造环境中人的部分脑力劳动。智能加工系统已经成为现代化生产的大趋势。

1.1.2 动态调度

动态调度通常是指在调度环境和任务存在不可预测扰动情况下所进行的调度。与静态调度相比，动态调度能够针对生产现场的实际情况产生更具可操作性的决策方案。实际生产中的大量问题是随机发生的。如在机械制造业中，由于工件随机到达，加工机器出现故障等随机事件，使得预调度不能正常执行，这就需要安排重调度。

1.1.3 研究意义

大部分离散制造企业属于多品种小批量的生产模式，特点为产品品种多、数量少、生产重复性小、工艺过程经常变更等。因此需要一个先进适用的管理系统，对于下达的生产任务进行一定程度上的智能优化调度，最大程度地减少生产过程中的非增值时间，实现加工生产的高效率、高柔性和高可靠性。

1.2 相关数据

1. 题目提取：见表1-1

表 **1-1**　智能加工系统数据

序号	名称	数量
1	计算机数控机床 (CNC)	8 台
2	轨道式自动引导车 (RGV)	1 辆
3	RGV 直线轨道	1 条
4	上料传送带	1 条
5	下料传送带	1 条

2. 智能加工系统作业参数的 3 组数据表（见原题表 1）

1.3 具体任务

1. 任务一

分别根据以下三种情况：一道工序的物料加工作业，两道工序的物料加工作业，考虑 CNC 以 1% 的概率发生故障，给出 RGV 动态调度模型和相应的求解算法。

2. 任务二

利用题目中表 1 给出的系统作业参数的 3 组数据，代入上述算法，给出 RGV 的调度策略和系统的作业效率。

§2 问题的分析

2.1 研究现状综述

国内外学者对 RGV 系统的优化做了一些研究，主要集中于 RGV 的派遣规则、路径选择及环轨多 RGV 系统的死锁避免等方面的仿真研究，但对 RGV 的单线往复动态调度问题的研究却几乎没有。

Dotoli 和 Fanti 应用着色赋时 Petri 网提出了 RGV 与巷道堆垛机的物料搬运系统的模块化建模框架[1]。Chen 等研究了 FMS 中的 RGV 调度和控制系统[2]。吴长庆等提出了基于双层着色赋时 Petri 网的 RGVs 系统的动态模型，并采用最短路径的调度策略[3]。陈华等针对自动化立体仓库出库过程中的直线往复两穿梭车 (RGV) 系统可能存在的 RGV 相互碰撞问题，提出了 RGV 冲突避免的约束条件[4]。杨少华等用排队论对环轨多车情况下的 RGV 数量和能力进行了分析[5]。吴焱明等使用仿真的方法研究了环轨 RGV 系统的动态调度问题，分析对比了两种 RGV 调度方法[6]。

总而言之，现有的文献或多或少都有其不足之处，需要加以完善。

2.2 对问题的总体分析和解题思路

本题需要研究的问题是智能 RGV 的动态调度方案的设计，包含加工一道工序的物料，加工两道工序的物料和考虑 CNC 在加工过程中出现故障时加工一道工序和加工两道工序四种情况，并据此对题目中表 1 的三组数据进行模型和算法的检验。针对此问题，我们分为四个小问题来进行研究。第一，利用 MATLAB 编程，对 CNC 加工一道工序物料的情况进行 RGV 的动态调度方案设计，并求解；第二，利用 MATLAB 编程，考虑工序分配，对 CNC 加工两道工序物料的情况进行 RGV 的动态调度方案设计，并求解；第三，针对 CNC 在加工过程中以 1% 的概率发生故障时加工一道工序物料的情况，在 10

到 20 分钟的误工时间后加入工作队列，重新设计算法，并求解；第四，针对 CNC 在加工过程中出现故障时加工两道工序物料的情况，改进前面的模型，并求解。

根据本文的研究思路，做出整体的思路流程图，如图2-1所示。

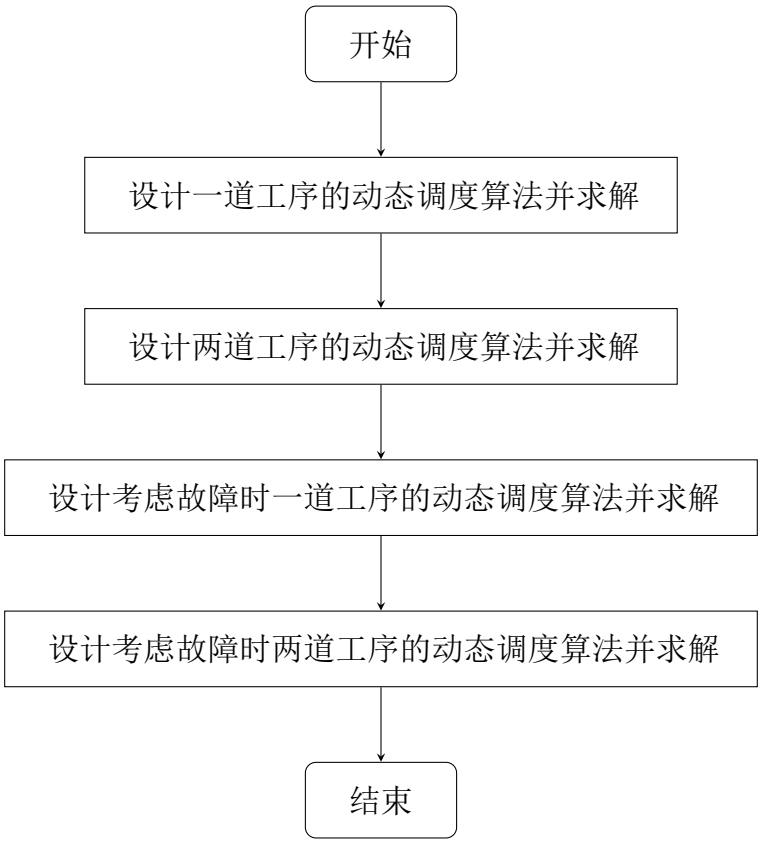

图 **2-1**　整体思路流程图

2.3 对具体任务的分析和对策

1. 对任务一的分析和对策

任务 1 要求对一般问题进行研究，给出智能 RGV 的动态调度方案的设计，包含加工一道工序的物料，加工两道工序的物料和考虑 CNC 在加工过程中出现故障时一道工序和两道工序四种情况。针对此问题，首先，利用 MATLAB 编程，基于贪心算法进行加工一道工序的物料时 RGV 的动态调度方案设计；其次，假设三种不同的工序分配方案：第一道工序与第二道工序各自 4 台 CNC，执行第一道工序的 CNC 数目大于执行第二道工序的 CNC 数目，执行第一道工序的 CNC 数目小于执行第二道工序的 CNC 数目，将一道工序的算法推广至两道工序，同样是基于贪心算法；最后，考虑 CNC 在加工过程中发生故障的情况，按照 1% 的概率随机设置 CNC 停止工作，在 10 到 20 分钟的误工时间后加入工作队列，误工时间由区间 [10, 20] 内的均匀分布产生。根据模拟退火算法的思想建立模型，跳出局部最优解，以寻找全局最优解。

2. 对任务二的分析和对策

任务 2 要求利用 3 组数据分别检验算法的有效性，给出 RGV 的调度策略和系统的作业效率，并将具体的结果分别填入附件 2 的 EXCEL 表中。在任务 1 中已经得到在有无故障两种情况下三种不同工序分配方案的 RGV 动态调度模型，将题目中表格 1 的数据分别带入相应模型的算法中，使用 MATLAB 求解，可得最终结果。

§3 模型的假设

1. 假设 CNC 在不工作时不会发生故障
2. 假设 RGV、传送带等设备在工作中不会发生故障
3. 假设物料供给及时且充足，不会出现断料的情况
4. 假设 RGV 内部程序的运行时间可以忽略不计
5. 假设 RGV 在加工两道工序的物料时，第一道工序完成后不清洗物料
6. 假设题目所给数据真实正确

§4 名词解释与符号说明

4.1 名词解释

1. 计算机数控机床

计算机数控机床 (CNC) 是一种装有程序控制系统的自动化机床。该控制系统能够逻辑地处理具有控制编码或其他符号指令规定的程序，并将其译码，用代码化的数字表示，通过信息载体输入数控装置。经运算处理由数控装置发出各种控制信号，控制机床的动作，按图纸要求的形状和尺寸，自动地将零件加工出来。

2. 轨道式自动引导车

轨道式自动引导车 (RGV) 可用于各类高密度储存方式的仓库以及智能加工车间，小车通道可设计为任意长度，这可提高整个仓库储存量，并且在操作时无需叉车驶入巷道，安全性更高。利用叉车无需进入巷道的优势，配合小车在巷道中的快速运行，可以有效提高仓库以及车间的运行效率。

3. 工序分配

本文中共有三种不同的工序分配方案：第一道工序与第二道工序各自 4 台 CNC；执行第一道工序的 CNC 数目大于执行第二道工序的 CNC 数目；执行第一道工序的 CNC 数目小于执行第二道工序的 CNC 数目。

4. 误工时间

CNC 在加工过程中会以 1% 的概率发生故障，在 10 到 20 分钟的时间后加入工作队列，这段耽误的时间称为误工时间，并且这段误工时间可由区间 [10, 20] 内的均匀分布、正态分布、泊松分布等产生，我们在这里选用均匀分布。

4.2 符号说明

表 **4-1** 符号说明

序号	符号	意义
1	$CNCi\#$	编号为 i 的 CNC, $i = 1, 2, \cdots, 8$
2	t_{mj}	RGV 移动 j 个单位所需时间, $j = 0, 1, 2, 3$
3	t_{cnc}	CNC 加工完成一个一道工序的物料所需时间
4	t_{cnc1}	CNC 加工完成一个两道工序物料的第一道工序所需时间
5	t_{cnc2}	CNC 加工完成一个两道工序物料的第二道工序所需时间
6	P	RGV 小车位置, $P = 1, 2, 3, 4$
7	P'	RGV 小车下一步要去的位置, $P' = 1, 2, 3, 4$
8	P^1	第一道工序包含的 CNC 编号集合
9	P^2	第二道工序包含的 CNC 编号集合
10	p	机械爪上是否有熟料标志, 1 表示有, 0 表示无
11	S_t	绝对理想的状态下, 系统加工获得的成料数量
12	S_m	模型计算出的系统加工获得的成料数量
13	S_i	工序分配集合, $i = 1, 2, 3$
14	η	基于模型结果与绝对理想结果得出的系统的效率
15	$\mathbf{T_{cncw}} \in \mathbb{R}^{1\times 8}$	CNC 剩余工作时间矩阵, 存储 CNC 的剩余工作时间
16	$\mathbf{T_{rgvm}} \in \mathbb{R}^{1\times 8}$	RGV 移动时间矩阵, 即 RGV 移动到下一位置所需时间
17	$\mathbf{T_{rgvw}} \in \mathbb{R}^{1\times 8}$	RGV 工作时间矩阵, 即 RGV 上下料的时间
18	$\mathbf{T_{crash}} \in \mathbb{R}^{1\times 8}$	一道工序情况下, 考虑故障时 CNC 剩余工作时间矩阵
19	$\mathbf{T'_{crash}} \in \mathbb{R}^{1\times 8}$	两道工序情况下, 考虑故障时 CNC 剩余工作时间矩阵

§5 模型的建立与求解

5.1 一道工序情况的分析与求解

5.1.1 对任务的分析

针对任务一，我们首先对 CNC 加工一道工序的物料情况进行建模，图5-1和图5-2从不同角度展示了整个智能加工系统。

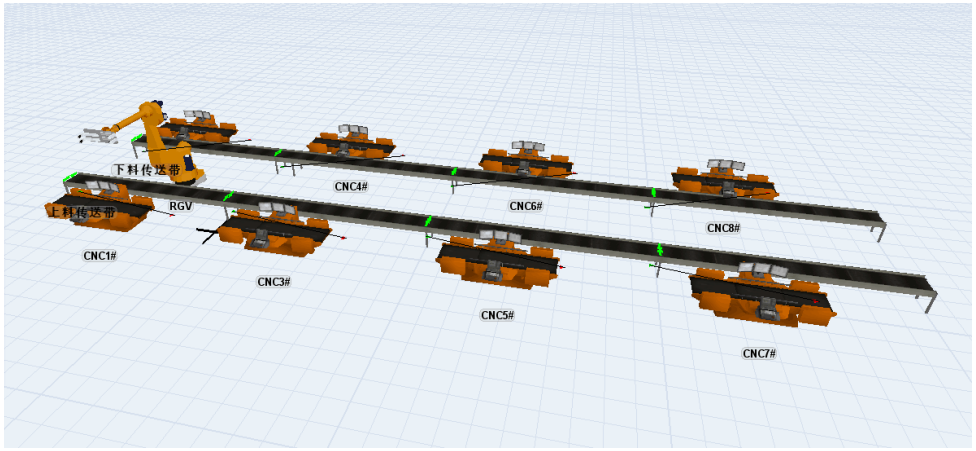

图 **5-1**　智能加工系统示意图

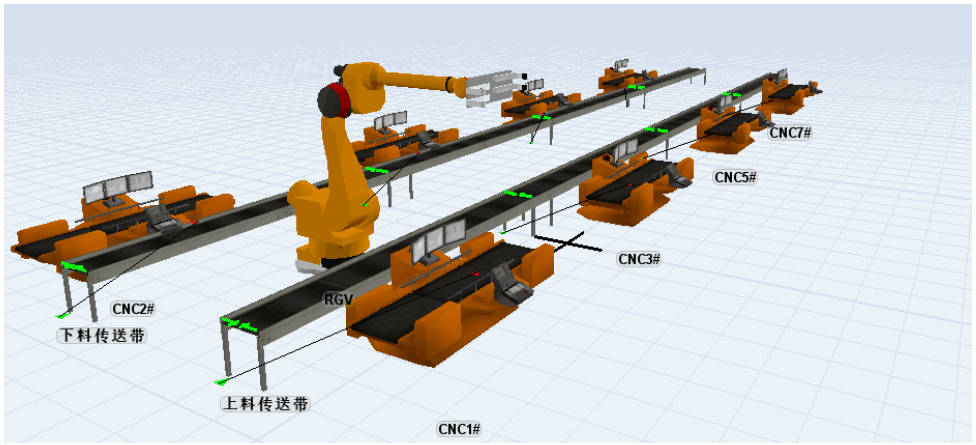

图 **5-2**　智能加工系统示意图

定义 RGV 的位置 P：

定义 1 *RGV 在智能加工系统中的位置 P 有四种情况，*

$$P=\begin{cases}1, & RGV\text{ 位于 }CNC1\#\text{ 与 }CNC2\#\text{ 正中间}\\ 2, & RGV\text{ 位于 }CNC3\#\text{ 与 }CNC4\#\text{ 正中间}\\ 3, & RGV\text{ 位于 }CNC5\#\text{ 与 }CNC6\#\text{ 正中间}\\ 4, & RGV\text{ 位于 }CNC7\#\text{ 与 }CNC8\#\text{ 正中间}\end{cases} \tag{5-1}$$

RGV 在某一位置工作完成后，下一个要去的最优的位置，是在所有可能的情况中行走时间与等待该位置 CNC 工作结束时间之和最小的位置，即贪心算法的主要思想，每一步只寻找当前最优解。

下面基于这个思想进行模型构建。

5.1.2 对任务的求解

模型 I—基于贪心算法求解最优动态调度方案模型

1. 模型的建立

根据前面的分析，设定 RGV 的动态调度原则 (模型) 如下：

原则 1 *RGV 下一步要去的位置 P' 为 RGV 从当前位置 $P=i$ 行走到 $P=j$ 的时间 $\mathbf{T_{rgvm}}(j)$ 与 $\min\{\mathbf{T_{cncw}}(2j-1)+\mathbf{T_{rgvw}}(2j-1), \mathbf{T_{cncw}}(2j)+\mathbf{T_{rgvw}}(2j)\}$ 之和最小的位置，即*

$$P'=j\Big|\min\left\{\mathbf{T_{rgvm}}(j)+\min\{\mathbf{T_{cncw}}(2j-1)+\mathbf{T_{rgvw}}(2j-1), \mathbf{T_{cncw}}(2j)+\mathbf{T_{rgvw}}(2j)\}\right\}, \quad (5\text{-}2)$$

这里, $i, j=1,2,3,4$,

$$\mathbf{T_{rgvm}}(j)=t_{m|j-i|} \quad (5\text{-}3)$$

为 RGV 移动时间矩阵中第 j 个元素, 即 RGV 移动到下一位置 j 所需时间;

$\mathbf{T_{cncw}}(2j-1)$, $\mathbf{T_{cncw}}(2j)$ 为 CNC 剩余工作时间矩阵中第 $2j-1$ 和 $2j$ 个元素, 即 $CNC(2j-1)\#$ 和 $CNC(2j)\#$(这两台 CNC 位于正对面) 的剩余工作时间.

$\mathbf{T_{rgvm}}(2j-1)$, $\mathbf{T_{rgvm}}(2j)$ 为 RGV 工作时间矩阵中第 $2j-1$ 和 $2j$ 个元素, 即 RGV 给 $CNC(2j-1)\#$ 和 $CNC(2j)\#$(这两台 CNC 位于正对面) 上下料的时间.

2. 模型的算法

一道工序情况下基于贪心算法求解最优动态调度方案模型的算法如流程图5-3所示。

RGV 动态调度的第一个模型主要用于解决一道工序的物料作业加工情况，且不考虑机器故障问题。该模型基于贪婪法在每次 RGV 要做出相应动作时求得执行下一步所需的最短时间并基于这一原则执行下一步的路径操作。

将每次要执行的操作所需时间分为三个部分：RGV 的移动、RGV 的工作和 CNC 剩余的工作时间，根据三个时间计算下一步的最短时间并进行相应路径选择。选择完毕后根据 RGV 下一步需要执行的操作来增加相应的操作时间，直到 8 小时结束时循环结束。若即将到达八小时，机器将不再上料，等待所有下料清洗结束后回到初始状态。此部分在程序中并未体现，需人工去除程序生成的最后几项超时的物料。

3. 模型的求解

(1) RGV 的调度策略

根据 MATLAB 求解的结果可得出第一组 RGV 动态调度情况，如图5-4所示，三组数据得出的动态调度情况类似，只是加工的成料数目不同，故只画出第一组的情况示意。

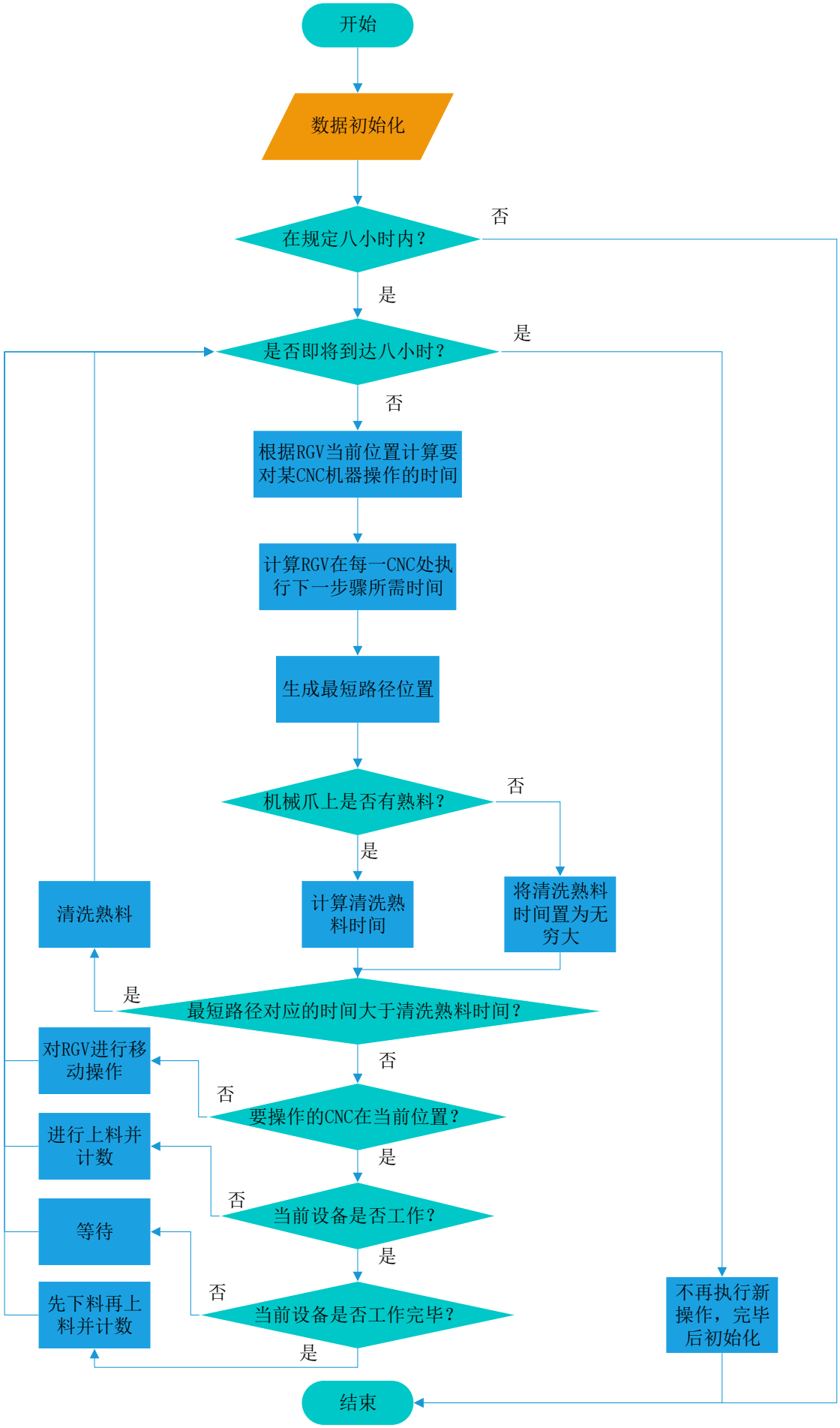

图 **5-3**　一道工序基于贪心算法模型的流程图

在加工一道工序的物料的情况下，RGV 首先按照 CNC1# 至 CNC8# 的顺序给各个 CNC 上料，然后回到 $P = 1$ 处等待，等到 CNC1# 至 8# 逐个加工完毕，RGV 按照 CNC1# 至 CNC8# 的顺序给各个 CNC 上下料，然后又回到 $P = 1$ 处等待。以后一直重复此循环。

RGV 的调度策略为每一步只寻找当前最优解，具体表述为原则 1。在加工一道工序物料的情况下，RGV 的调度策略简化为按照 CNC1# 至 CNC8# 的顺序循环工作，直至工作结束。

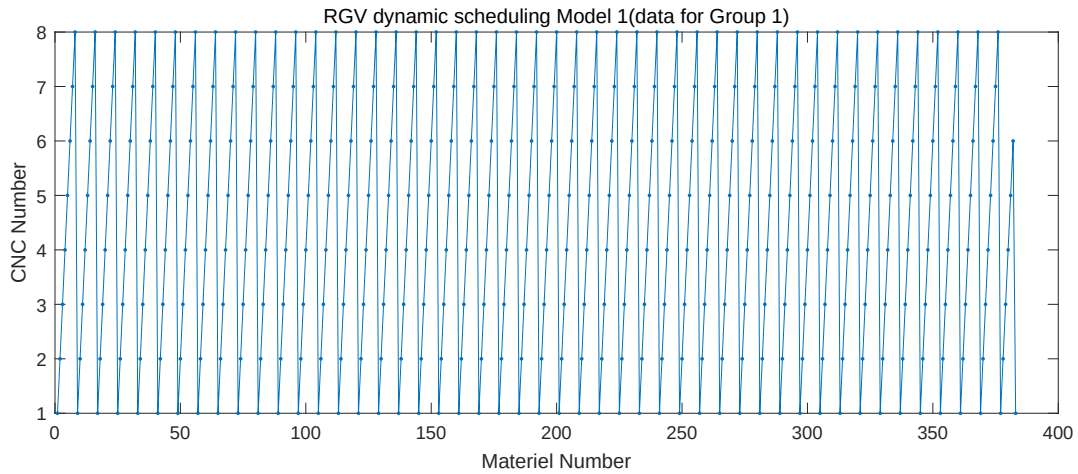

图 **5-4**　一道工序情况下 **RGV** 动态调度示意图

(2) 系统的作业效率

在绝对理想的状态下，8 个小时内 8 台 CNC 永远在工作，没有停歇时间。此时加工物料的总量为：

$$S_t = \frac{8 \times 60 \times 60 \times 8}{t_{cnc}} \tag{5-4}$$

代入 3 组数据的 t_{cnc} 可得：

$$S_{t1} = 411,\ S_{t2} = 397,\ S_{t3} = 422.$$

由模型 I 计算出的结果，可知一道工序情况下 8 个小时加工物料的总量为：

$$S_{m1} = 382,\ S_{m2} = 358,\ S_{m3} = 391.$$

系统作业效率 η 为：

$$\eta = \frac{S_m}{S_t} \times 100\%, \tag{5-5}$$

代入以上数据可得：

$$\eta_1 = 92.94\%,\ \eta_2 = 90.18\%,\ \eta_3 = 92.65\%.$$

(3) 模型的实用性和算法的有效性

通过对 MATLAB 输出的数据进行分析可知，RGV 是按照 CNC1# 至 CNC8# 的顺序循环工作的，易知这种状态就是整个系统能够达到的最优状态。所以，虽然我们是通过每次寻找局部最优的方法入手解决问题，但是我们得到的结果已经是全局最优解。

经 3 组数据检验，模型得出了合理的结论，具有实用性。算法快速有效，系统的作业效率约为 92%，已经达到了全局最优。

5.2 两道工序情况的分析与求解

5.2.1 对任务的分析

两道工序的物料加工作业情况，每个物料的第一和第二道工序分别由两台不同的 CNC 依次加工完成。针对此问题，我们首先要确定加工第一道工序的 CNC 分别是哪几台，同时确定加工第二道工序的 CNC 是哪几台。RGV 首先对加工第一道工序的 CNC 上料，等待第一道工序结束，将半成品上料到加工第二道工序的 CNC，然后大致重复此流程。RGV 具体的动态调度采取寻找当前最优解的贪心算法策略。

5.2.2 对任务的求解

模型 II—基于贪心算法求解最优动态调度方案模型

1. 模型的建立

我们首先进行工序分配，针对三组不同的数据，指定三种不同的工序分配方案：

$$\begin{cases} 1: & \text{第一道工序与第二道工序各自 4 台 CNC} \\ 2: & \text{执行第一道工序的 CNC 数目大于执行第二道工序的 CNC 数目} \\ 3: & \text{执行第一道工序的 CNC 数目小于执行第二道工序的 CNC 数目} \end{cases}$$

定义工序分配集合：

定义 2 *三种不同的工序分配方案对应的工序分配集合定义为*

$$S_1 = \{(4,4)\},\ S_2 = \{(5,3),(6,2),(7,1)\},\ S_3 = \{(3,5),(2,6),(1,7)\},$$

其中每种情况用一个数对表示，第一个数字表示执行第一道工序的 CNC 数目，第二个数字表示执行第二道工序的 CNC 数目。

针对三种不同的工序分配方案，我们对 $S_i,\ i = 1,2,3,$ 循环遍历所有可能的情况，确定哪几台 CNC 执行第一道工序，哪几台 CNC 执行第二道工序，可以使系统的生产效率达到最优，一共遍历了 $C_8^1 + C_8^2 + \cdots + C_8^7 = 2^8 - 2 \times 1 = 254$ 种情况。

虽然工序分配有三种情况，但是 RGV 的动态调度原则可以统一表述。记第一道工序包含的 CNC 编号集合为 P^1, 第二道工序包含的 CNC 编号集合为 P^2, 那么，两道工序情况下 RGV 的动态调度原则 (模型) 为：

原则 2 *RGV 下一步要去的位置 P' 为 RGV 从当前位置 $P = i$ 行走到 $P = j$ 的时间 $\mathbf{T}_{\mathbf{rgvm}}(j)$ 与该位置两台 CNC 之一加工剩余时间以及 RGV 给其上下料时间之和最小的位置。p 为*

机械爪上是否有熟料标志, 1 表示有，0 表示无。当 $p=0$ 时，*RGV* 只处理第一道工序 *CNC* 发出的请求；当 $p=1$ 时，*RGV* 只处理第二道工序 *CNC* 发出的请求。那么，*RGV* 下一步要去的位置 P' 为

$$P'=\begin{cases} j\left|\min\begin{cases}\min\left\{\mathbf{T_{rgvm}}(j)+\mathbf{T_{cncw}}(2j-1)+\mathbf{T_{rgvw}}(2j-1)\right\}, & if\ 2j-1\in P^1,\\ \min\left\{\mathbf{T_{rgvm}}(j)+\mathbf{T_{cncw}}(2j)+\mathbf{T_{rgvw}}(2j)\right\}, & if\ 2j\in P^1,\end{cases}\right., & if\ p=0;\\ j\left|\min\begin{cases}\min\left\{\mathbf{T_{rgvm}}(j)+\mathbf{T_{cncw}}(2j-1)+\mathbf{T_{rgvw}}(2j-1)\right\}, & if\ 2j-1\in P^2,\\ \min\left\{\mathbf{T_{rgvm}}(j)+\mathbf{T_{cncw}}(2j)+\mathbf{T_{rgvw}}(2j)\right\}, & if\ 2j\in P^2,\end{cases}\right., & if\ p=1. \end{cases} \tag{5-6}$$

2. 模型的算法

在两道工序的情况下，基于贪心算法求解最优动态调度方案模型的算法如流程图5-6所示。

RGV 动态调度的第二个模型主要用于解决不考虑机器故障情况下的两道工序物料作业加工情况。该模型基于贪婪法在每次 RGV 要执行下一动作时求得对每个设备执行的时间长短并以此来作出选择。由于出现了两道工序，所以最开始需要对所有分组情况进行遍历，得到最大物料数的分组即为最优解。

由于不同的 CNC 分成了两组，因此通过更改总时间矩阵某些项的大小可以达到先进行第一道工序再进行第二道工序的效果。分组完毕后，根据 RGV 下一步需要执行的操作来增加相应的操作时间。若即将到达八小时，机器将不再完成上料工作，等待所有下料清洗结束后回到初始状态。此部分在程序中并未体现，需人工去除程序生成的最后几项超时的物料。

3. 模型的求解

(1) RGV 的调度策略

通过对 S_i, $i=1,2,3$, 中所有可能情况的遍历，最终确定第一组数据应该采用工序分配方案为 (4,4)，并且取得最优解的 (4,4) 的具体分配不唯一，其中一种可行的方案为：第一道工序为 CNC1#, CNC3#, CNC5#, CNC7#；第二道工序为 CNC2#, CNC4#, CNC6#, CNC8#。

第二组应该采用的工序分配方案为 (4,4)，并且取得最优解的 (4,4) 的具体分配只有一种：第一道工序为 CNC2#, CNC4#, CNC6#, CNC8#；第二道工序为 CNC1#, CNC3#, CNC5#, CNC7#。

而第三组应该采用的工序分配方案为 (5,3)，并且取得最优解的 (5,3) 的具体分配不唯一，其中一种可行的方案为：第一道工序为 CNC1#, CNC3#, CNC4#, CNC6#, CNC8#；第二道工序为 CNC2#, CNC5#, CNC7#。

RGV 的调度策略与模型 I 类似，每一步只寻找下一步的局部最优解，具体表述为原则 2。由 MATLAB 作出第一组数据的 RGV 动态调度示意图如5-7所示，其他两组数据的图片放在附录。

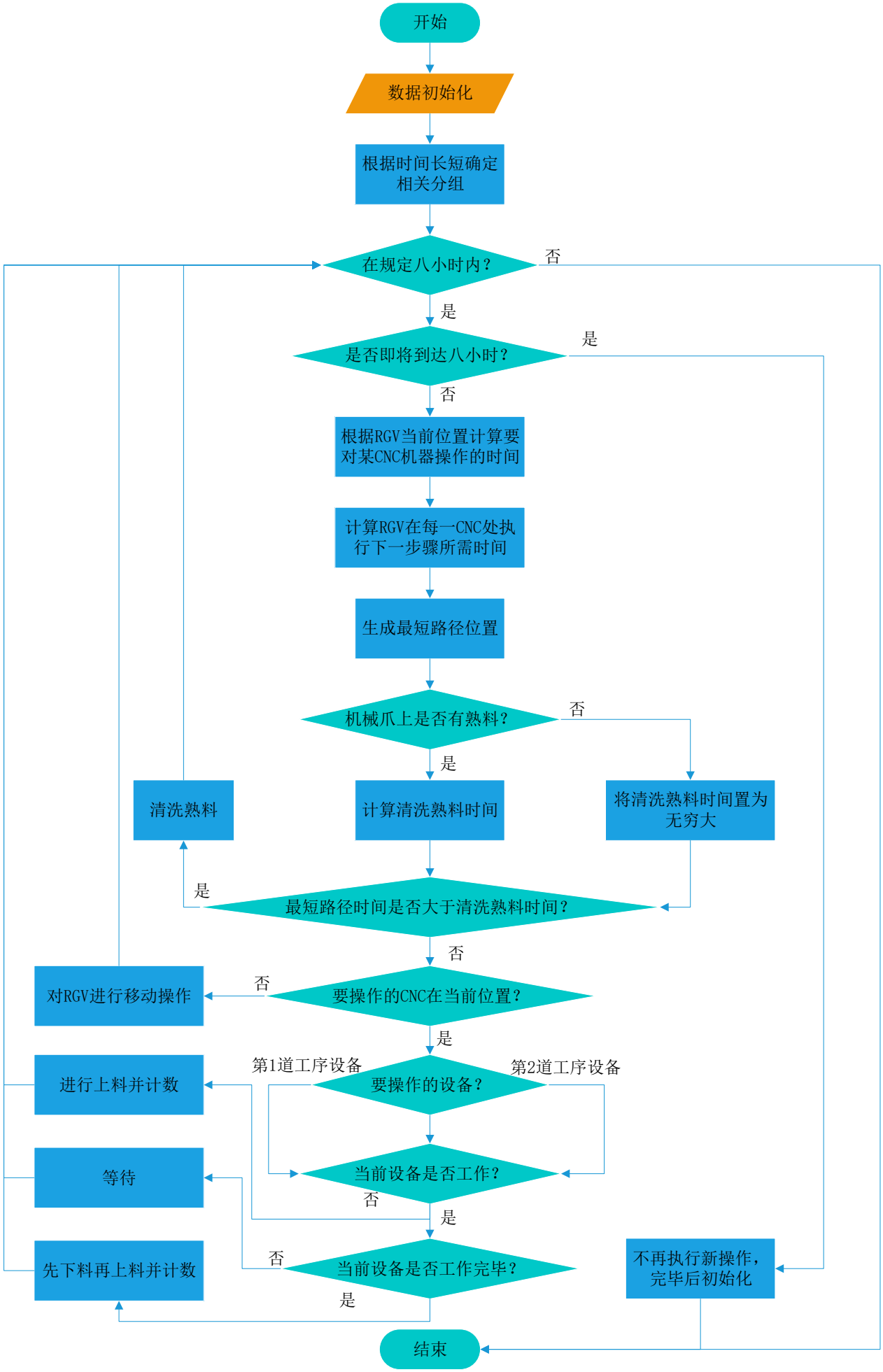

图 **5-5**　两道工序基于贪心算法模型的流程图

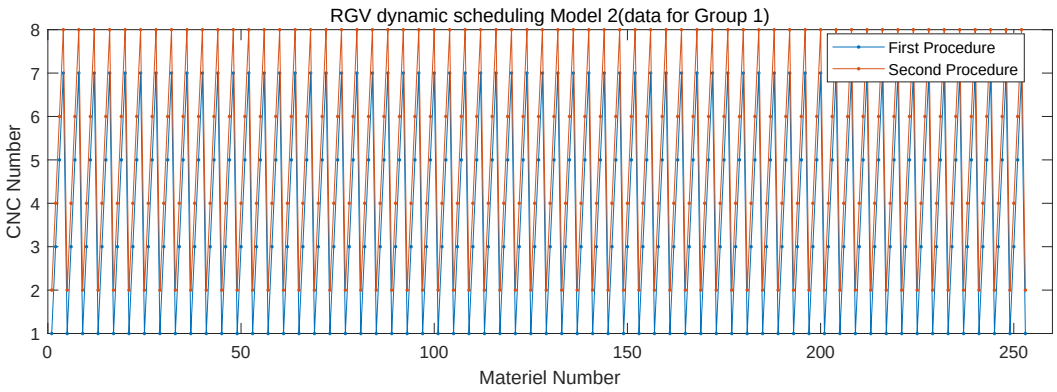

图 **5-6**　两道工序情况下 **RGV** 动态调度示意图 **(**第一组数据**)**

(2) 系统的作业效率

在绝对理想的状态下，8 个小时内加工第二道工序的 CNC 永远在工作，没有停歇时间。代入 3 组数据可得此时加工物料的总量为：

$$S_{t1}=\frac{28800\times 4}{t_{cnc2}}=288,\ S_{t2}=\frac{28800\times 4}{t_{cnc2}}=230,\ S_{t3}=\frac{28800\times 3}{t_{cnc2}}=274.$$

由模型 II 计算出的结果为：

$$S_{m1}=253,\ S_{m2}=210,\ S_{m3}=245.$$

系统作业效率 η 为：

$$\eta_1=87.85\%,\ \eta_2=91.30\%,\ \eta_3=89.42\%.$$

(3) 模型的实用性和算法的有效性

通过对模型结构以及 MATLAB 输出的数据进行分析可知，模型对 3 组数据分别给出了相对的最优解，具有实用性。算法快速有效，系统的作业效率约为 89%，已经达到了比较理想的状态。

5.3 考虑故障概率一道工序与两道工序情况的分析与求解

5.3.1 对任务的分析

CNC 在加工过程中以大约 1% 的概率发生故障，发生故障时，未完成的物料报废。误工时间介于 10 到 20 分钟之间，故障排除后立即加入作业序列。在有 CNC 发生故障的情况下，RGV 需要重新进行动态调度的规划，在剩余的没有发生故障的 CNC 中寻求最优位置。我们采用模拟退火算法的思想对这种情况进行建模求解。

5.3.2 对任务的求解

模型 III—基于模拟退火算法求解最优故障动态调度方案模型：一道工序

1. 模型的建立

CNC 在加工过程中以大约 1% 的概率发生故障的情况下，RGV 的动态调度原则与没有故障时相似。为了给出统一的表达式，我们首先定义一个考虑故障时一道工序情况下 CNC 的剩余工作时间矩阵：

定义 3 *一道工序情况下，考虑故障时 CNC 剩余工作时间矩阵*

$$\mathbf{T_{crash}}(i)=\begin{cases}\mathbf{T_{cncw}}(i), & \text{若 } CNCi\# \text{ 没有发生故障,}\\ t_{cnc}, & \text{若 } CNCi\# \text{ 发生故障,}\end{cases},\ i=1,2,\cdots,8. \tag{5-7}$$

当 CNCi# 发生故障时，将 $\mathbf{T_{crash}}(i)$ 取为 CNC 加工完成一个一道工序的物料所需时间 t_{cnc}, 可以在取最小值时避免取到发生故障的 CNCi#.

这样我们得到了考虑故障时 CNC 的剩余工作时间矩阵 $\mathbf{T_{crash}}$，将原则 1 中 $\mathbf{T_{cncw}}$ 换为 $\mathbf{T_{crash}}$ 即可确定考虑故障时在一道工序情况下 RGV 的动态调度原则 (模型)。

原则 3 *RGV 下一步要去的位置 P' 为 RGV 从当前位置 $P=i$ 行走到 $P=j$ 的时间 $\mathbf{T_{rgvm}}(j)$ 与 $\min\{\mathbf{T_{crash}}(2j-1)+\mathbf{T_{rgvw}}(2j-1),\mathbf{T_{crash}}(2j)+\mathbf{T_{rgvw}}(2j)\}$ 之和最小的位置，即*

$$P'=j\Big|\min\left\{\mathbf{T_{rgvm}}(j)+\min\{\mathbf{T_{crash}}(2j-1)+\mathbf{T_{rgvw}}(2j-1),\mathbf{T_{crash}}(2j)+\mathbf{T_{rgvw}}(2j)\}\right\},\ i,j=1,2,3,4. \tag{5-8}$$

2. 模型的算法

考虑故障时，一道工序情况下基于模拟退火算法求解最优动态调度方案模型的算法如流程图5-8所示。

RGV 动态调度的第三个模型主要用于解决可能发生机器故障的一道工序物料作业加工情况。该模型利用贪婪法求出 RGV 下一步进行操作的最短路径，为了保证可以找到更加优化的解，该模型在生成最短路径时采用了模拟退火算法，一定概率寻找倒数第二短的路径并执行相应操作。

根据对总时间矩阵的判定选择路径。选择完毕后根据 RGV 下一步需要执行的操作来增加相应的操作时间。每给机器上一次料就进行一次故障检测，若无故障则按照原来的规则继续运行，若有故障，则生成一个故障发生的时间以及人工维护需要的时间 (利用均匀分布)，然后重新计算总时间矩阵选择最优路径。若即将到达八小时，机器将不再完成上料工作，等待所有下料清洗结束后回到初始状态。此部分在程序中并未体现，需人工去除程序生成的最后几项超时的物料。

3. 模型的求解

(1) RGV 的调度策略

RGV 的调度策略为：通过模拟退火算法每一步基于原则 3 的思想寻找当前最优解，并且有一定概率寻找倒数第二短的路径，多次运行以寻求全局最优解，然后 RGV 按照

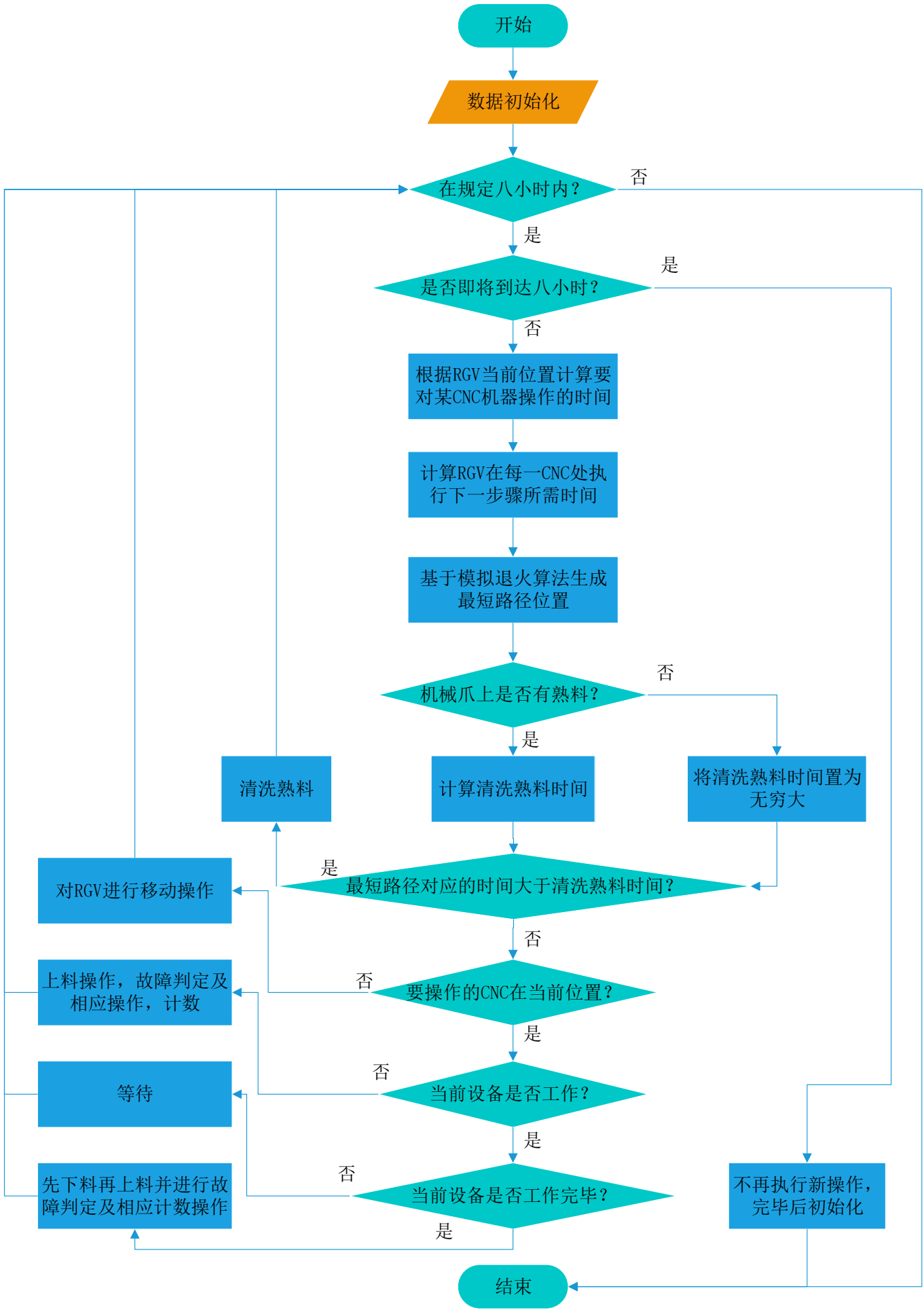

图 **5-7**　考虑故障时一道工序模拟退火模型的流程图

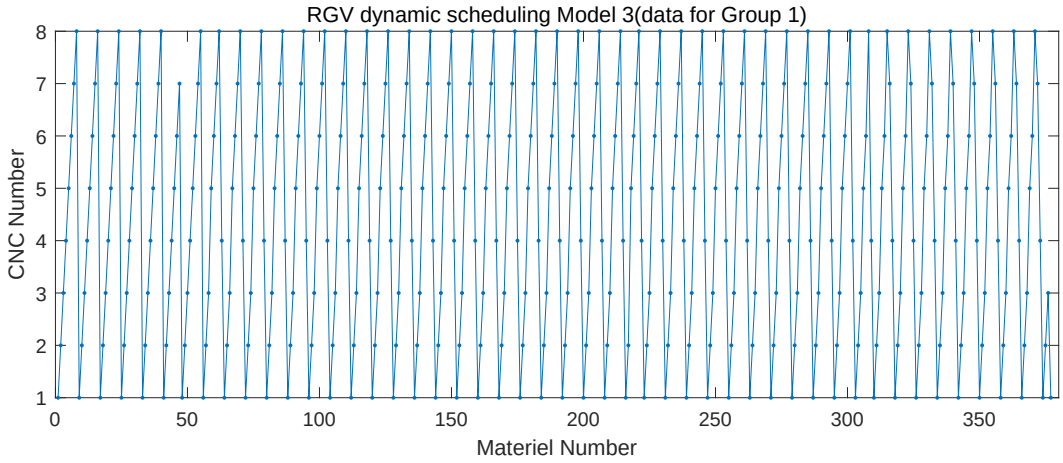

图 **5-8**　考虑故障时一道工序情况下 **RGV** 动态调度示意图 **(**第一组数据**)**

最优解的路线移动。由 MATLAB 作出第一组数据的 RGV 动态调度示意图如5-9所示，其他两组数据的图片放在附录。

(2) 系统的作业效率

在绝对理想的状态下，8 个小时内加工第二道工序的 CNC 永远在工作，不会发生故障，没有停歇时间。根据模型 I 中得出的公式 (5-4) 可知此时加工物料的总量与模型 I 相同，为：

$$S_{t1} = 411,\ S_{t2} = 397,\ S_{t3} = 422.$$

由模型 III 计算出的结果，可知一道工序情况下考虑故障时 8 个小时加工物料的总量为：

$$S_{m1} = 372,\ S_{m2} = 343,\ S_{m3} = 385.$$

系统作业效率 η 为：

$$\eta_1 = 90.51\%,\ \eta_2 = 86.40\%,\ \eta_3 = 91.23\%.$$

(3) 模型的实用性和算法的有效性

经数据检验，模型得出的结果基本符合有故障情况下智能加工系统的工作状态，但是由于试验样本较小，故障概率并未完全符合 1%。总的来说，模型合理地模拟了智能加工系统中有 CNC 会发生故障的情况，具有实用性。算法快速有效，较好的实现了有故障发生时 RGV 的动态调度，系统的作业效率约为 89%，已经达到了比较理想的状态。

模型 IV—基于模拟退火算法求解最优故障动态调度方案模型：两道工序

1. 模型的建立

在模型 III 中已经寻找到针对三组不同数据的最佳工序分配方案，模型 IV 延续此方案并对模型 III 的思想进行推广。

定义一个考虑故障时两道工序情况下 CNC 的剩余工作时间矩阵 $\mathbf{T}'_{\mathbf{crash}}$：

定义 4 两道工序情况下，考虑故障时 *CNC* 剩余工作时间矩阵

$$\mathbf{T}'_{\mathbf{crash}}(i)=\begin{cases}\mathbf{T}_{\mathbf{cncw}}(i), & \text{若 } CNCi\# \text{ 没有发生故障,}\\ t_{cnc}1, & \text{若 } CNCi\# \text{ 发生故障, 且 } i\in P^1,\quad i=1,2,\cdots,8.\\ t_{cnc}2, & \text{若 } CNCi\# \text{ 发生故障, 且 } i\in P^2,\end{cases}\tag{5-9}$$

当 *CNCi#* 发生故障时，将 $\mathbf{T}_{\mathbf{crash}}(i)$ 取为 *CNC* 加工完成一个第一道工序的物料所需时间 $t_{cnc}1$ 或加工完成一个第二道工序的物料所需的时间 $t_{cnc}2$, 可以在取最小值时避免取到发生故障的 *CNCi#.*

考虑故障时，两道工序情况下 RGV 的动态调度原则 (模型) 可以建立如下：

原则 4 *RGV* 下一步要去的位置 P' 为 *RGV* 从当前位置 $P=i$ 行走到 $P=j$ 的时间 $\mathbf{T}_{\mathbf{rgvm}}(j)$ 与该位置两台 *CNC* 之一加工剩余时间以及 *RGV* 给其上下料时间之和最小的位置。p 为机械爪上是否有熟料标志, 1 表示有，0 表示无。当 $p=0$ 时，*RGV* 只处理第一道工序 *CNC* 发出的请求；当 $p=1$ 时，*RGV* 只处理第二道工序 *CNC* 发出的请求。那么，*RGV* 下一步要去的位置 P' 为

$$P'=\begin{cases}j\left|\min\left\{\begin{array}{ll}\min\left\{\mathbf{T}_{\mathbf{rgvm}}(j)+\mathbf{T}_{\mathbf{crash}}(2j-1)+\mathbf{T}_{\mathbf{rgvw}}(2j-1)\right\}, & if\ 2j-1\in P^1,\\ \min\left\{\mathbf{T}_{\mathbf{rgvm}}(j)+\mathbf{T}_{\mathbf{crash}}(2j)+\mathbf{T}_{\mathbf{rgvw}}(2j)\right\}, & if\ 2j\in P^1,\end{array}\right\}, & if\ p=0;\\ j\left|\min\left\{\begin{array}{ll}\min\left\{\mathbf{T}_{\mathbf{rgvm}}(j)+\mathbf{T}_{\mathbf{crash}}(2j-1)+\mathbf{T}_{\mathbf{rgvw}}(2j-1)\right\}, & if\ 2j-1\in P^2,\\ \min\left\{\mathbf{T}_{\mathbf{rgvm}}(j)+\mathbf{T}_{\mathbf{crash}}(2j)+\mathbf{T}_{\mathbf{rgvw}}(2j)\right\}, & if\ 2j\in P^2,\end{array}\right\}, & if\ p=1.\end{cases}\tag{5-10}$$

2. 模型的算法

考虑故障时，两道工序情况下基于模拟退火算法求解最优动态调度方案模型的算法如流程图5-10所示。

RGV 动态调度的第四个模型主要用于解决可能发生机器故障情况下的两道工序物料作业加工情况。该模型利用贪心算法求出 RGV 下一步进行操作的最短路径。由于机器有损坏的可能，因此为了找到更优化的解，该模型在生成最短路径时采用了模拟退火算法，一定概率寻找倒数第二短的路径并执行相应操作。由第二个模型的结果可以得到对第四个模型三种情况的最佳分组。

由于不同的 CNC 分成两道工序，因此需要对总时间矩阵根据不同情况作出相应的修改以符合模型的实际情况。当得出下一步需要进行的操作以及操作位置时，根据相应的操作来增加相应的操作时间，当给一个机器（无论是第一道工序的机器还是第二道工序的机器）进行上料操作时，对这个机器进行一次故障检测，若有故障，则通过一定的概率模型生成此次故障发生的具体时间以及人工需要维护的时间（利用均匀分布），然后重新计算总时间矩阵选择最优路径。若即将到达八小时，机器将不再完成上料工作，

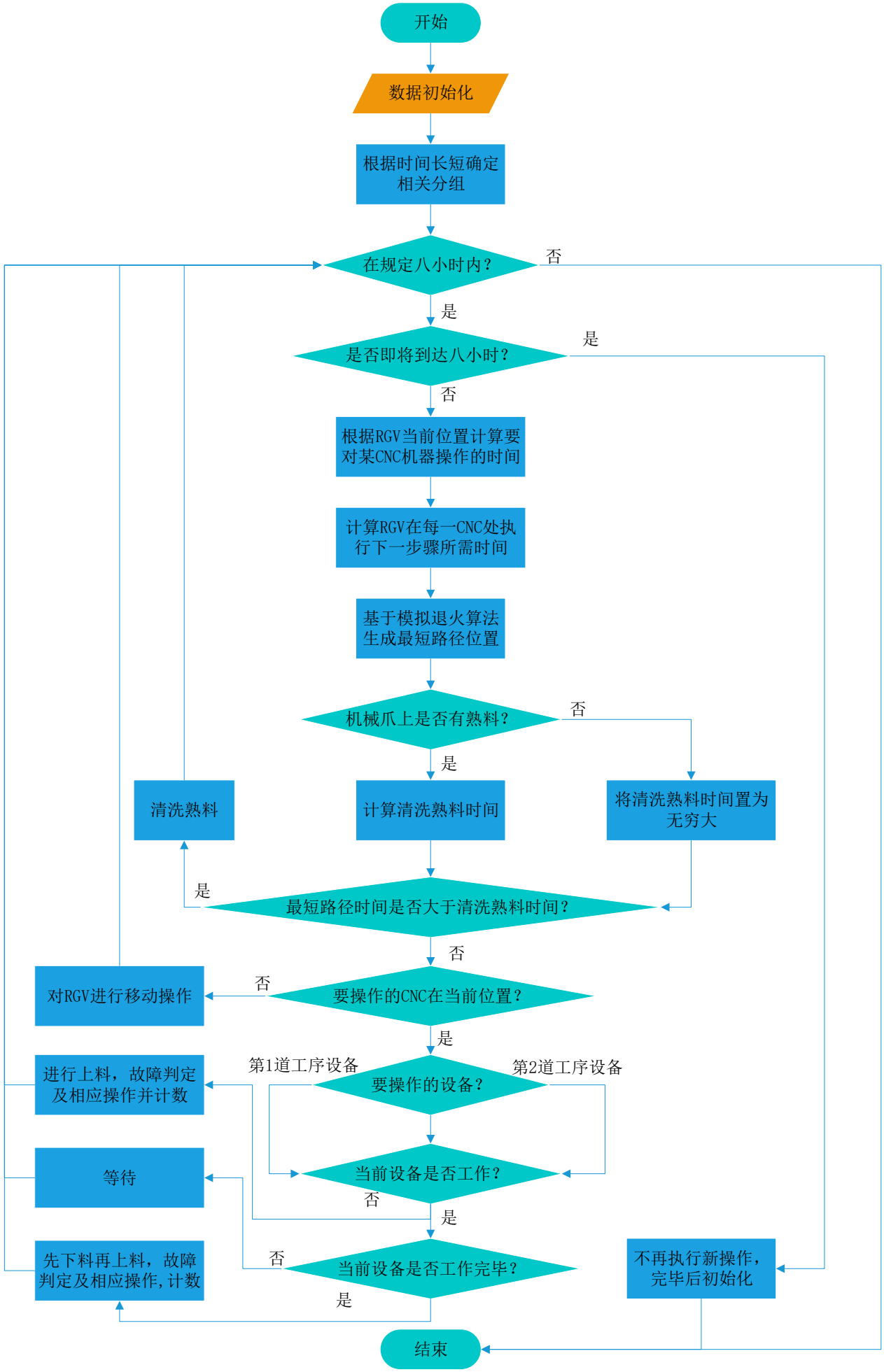

图 **5-9**　考虑故障时两道工序模拟退火模型的流程图

等待所有下料清洗结束后回到初始状态。此部分在程序中并未体现，需人工去除程序生成的最后几项超时的物料。

3. 模型的求解

(1) RGV 的调度策略

RGV 的调度策略为：通过模拟退火算法，每一步基于原则 4 的思想寻找当前最优解，并且有一定概率寻找倒数第二短的路径，多次运行以寻求全局最优解，然后 RGV 按照最优解的路线移动。由 MATLAB 作出第一组数据的 RGV 动态调度示意图如5-11所示，其他两组数据的图片放在附录。

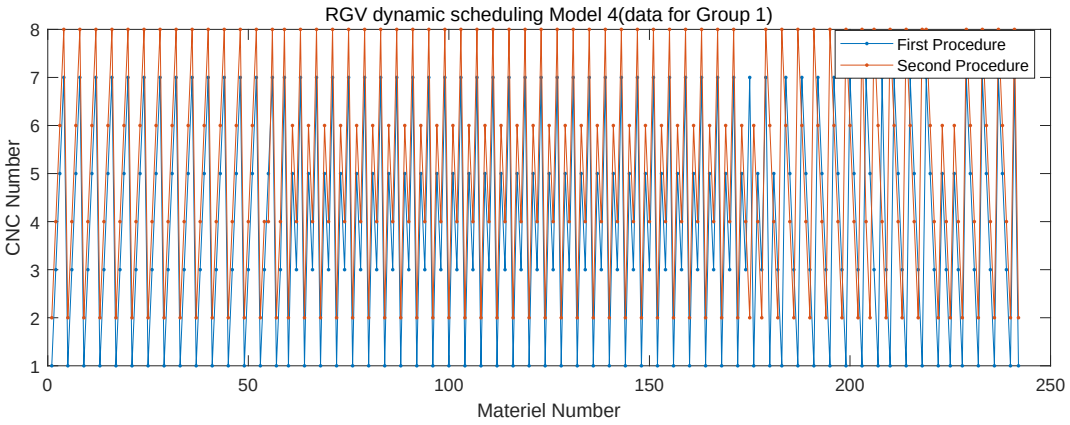

图 **5-10**　考虑故障时两道工序情况下 **RGV** 动态调度示意图 (第一组数据)

(2) 系统的作业效率

在绝对理想的状态下，8 个小时内加工第二道工序的 CNC 永远在工作，没有停歇时间。代入 3 组数据可得此时加工物料的总量为：

$$S_{t1} = \frac{28800 \times 4}{t_{cnc2}} = 288,\ S_{t2} = \frac{28800 \times 4}{t_{cnc2}} = 230,\ S_{t3} = \frac{28800 \times 3}{t_{cnc2}} = 274.$$

由模型 IV 计算出的结果，可知考虑故障时两道工序情况下 8 个小时加工的物料总量为：

$$S_{m1} = 237,\ S_{m2} = 198,\ S_{m3} = 238.$$

系统作业效率 η 为：

$$\eta_1 = 82.29\%,\ \eta_2 = 86.09\%,\ \eta_3 = 86.86\%.$$

(3) 模型的实用性和算法的有效性

经数据检验，模型合理地模拟了智能加工系统中考虑 CNC 会发生故障时两道工序的情况，具有实用性。算法快速有效，较好的实现了有故障发生时两道工序情况下 RGV 的动态调度，系统的作业效率约为 85%，已经达到了比较理想的状态。

§6 模型的仿真

FlexSim 是一个基于 Windows 的，面向对象的仿真环境，用于建立离散事件流程，例如制造业，物料处理和办公室工作流，这些全都配以相似度极高的三维虚拟现实环境。

由于时间有限，我们使用 FlexSim 只对本问题不考虑故障时一道工序的情况进行仿真，仿真中间过程如图6-1所示。

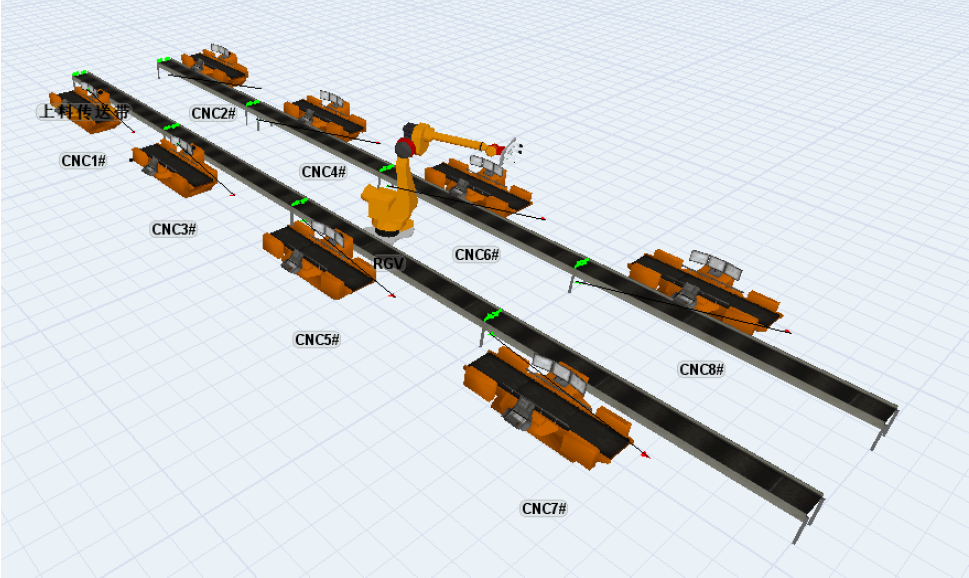

图 **6-1**　智能加工系统仿真模拟图

得到的结果与用 MATLAB 计算模型 I 得出的结果一致。

§7 误差分析与灵敏度分析

7.1 误差分析

1. 模型 III 和模型 IV 采用模拟退火算法寻找全局最优解，但是概率值的设定会对是否能够找到最优解有较大的影响，所以我们得到的不一定是全局最优解。即我们给出的结果与最优解之间可能存在误差。

2. CNC 发生故障后会停止工作 10 至 20 分钟，称为误工时间。我们对这段时间的处理是采用均匀分布随机生成一个 10 到 20 分钟的数作为误工时间，然而实际情况下，误工时间的分布可能是正态分布，泊松分布，超几何分布等。误工时间与实际情况可能存在误差。

7.2 灵敏度分析

将题目中所给 3 组数据中两道工序的时间比例进行扩展分析，设定两道工序总时长为 800 秒，分别设定五个不同的比例：200:600, 300:500, 400:400, 500:300, 600:200, 对模

型进行灵敏度分析，如图7-1所示。图中，横坐标 1 表示工序分配方案为 (4,4)，2 表示工序分配方案为 (5,3)，3 表示工序分配方案为 (3,5)，可以发现不同的两道工序时间比例对应的最佳工序分配方案可能会有差异，模型对此反应很灵敏。

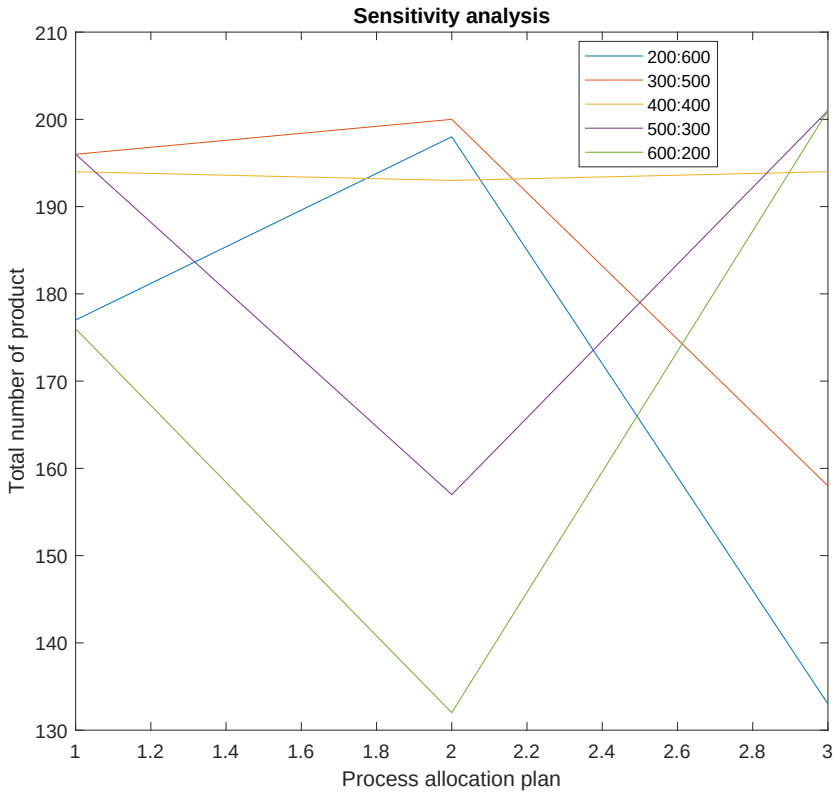

图 **7-1**　灵敏度分析

§8 模型的评价与推广

8.1 模型的评价

8.1.1 模型的优点

1、模型 I 以非常简单的贪心算法的思想就寻找到了全局最优解，简捷高效

2、模型 II 创新性地设置了机械爪上是否有熟料的标志变量，使得模型的数学表达式简洁优美；并且模型 II 遍历了所有可能的工序分配方案，可以处理所有不同的工序时间比例情况，具有通用性和普适性

3、模型 III 创新性地定义了一道工序情况下考虑故障时 CNC 剩余工作时间矩阵，使得模型的数学表达式非常简洁；并且模型 III 采用了模拟退火算法以弥补贪心算法可能陷入局部最优解的缺点，使得寻找到全局最优解的可能性大大提高。

4、模型 IV 创新性地定义了两道工序情况下考虑故障时 CNC 剩余工作时间矩阵，

使得模型 IV 的表达式与模型 III 可以统一为一个简洁的形式；并且模型 IV 对故障的模拟非常符合实际情况，使得模型的实用性大大增强。

8.1.2 模型的缺点

1、由于每班次连续作业 8 小时，作业结束时 RGV 需要回到原位，并且不能有未结束加工的 CNC，所以模型需要进行作业结束处理。这里我们采用人工对最后不能完成加工的 CNC 提前进行关机处理，没有完全实现自动化控制。

2、模型采用的模拟退火算法不能保证一定能够寻找到全局最优解。

8.2 模型的推广

1. 基于贪心算法求解最优动态调度方案模型的推广

基于贪心算法，我们设计了求解不考虑故障时，一道工序与两道工序情况下 RGV 的动态调度模型。事实上，该模型还可以解决多道工序情况下贪心算法还可以解决 RGV 的动态调度问题，以及类似情境下如仓库中往返存取货物等问题。并且，贪心算法还可以用来解决单源最短路径问题，最小生成树问题等。

2. 基于模拟退火算法求解最优故障动态调度方案模型的推广

基于模拟退火算法，我们设计了考虑故障时，一道工序与两道工序情况下 RGV 的动态调度模型。事实上，该模型还可以推广到广泛应用于自动化立体仓库系统中搬运物料的穿梭车上。并且，模拟退火算法还可以用于 VLSI 设计、图像识别和神经网计算机的研究等。

§9 模型的改进

本文在寻找最优路径的时候使用了模拟退火算法，但是模拟退火算法不能保证一定能够寻找到全局最优解。全局优化算法又称现代启发式算法，是一种具有全局优化性能、通用性强且适合于并行处理的算法。全局优化算法不止有模拟退火算法，还有遗传算法、禁忌搜索算法、粒子群算法、蚁群算法等。为了增大寻找到全局最优解的概率，可以使用不同的全局优化算法分别建模计算，比较所得结果，选取最优化的动态调度方案。

参考文献

[1] Dotoli M, Fanti M P. A colored Petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: a control perspective [J]. International Journal of Computer Integrated Manufacturing, 2005, 18(2-3): 122-136.

[2] Chen F F, Huang J, Centeno M.A. Intelligent scheduling and control of rail-guided vehicles and load/unload operations in a flexible manufacturing system[J]. Journal of Intelligent Manufacturing, 1999, 10(5): 405-421.

[3] 吴长庆, 罗键, 陈火国等. 基于 Petri 网的 RGV 系统中环路死锁研究 [J]. 计算机科学, 2009, 36(4): 250-253.

[4] 陈华, 孙启元. 基于 TS 算法的直线往复 2-RGV 系统调度研究 [J]. 工业工程与管理, 2015, 20(05): 80-88.

[5] 杨少华, 张家毅, 赵立. 基于排队论的环轨多车数量与能力分析 [J]. 制造业自动化, 2011, 33(16): 102-104.

[6] 吴焱明, 刘永强, 张栋等. 基于遗传算法的 RGV 动态调度研究 [J]. 起重运输机械, 2012(06): 20-23.

[7] 卓金武, 李必文, 魏永生等. MATLAB 在数学建模中的应用 [M], 北京：北京航空航天大学出版社, 2014.

[8] 龚纯, 王正林. 精通 MATLAB 最优化计算 [M], 北京: 电子工业出版社, 2012.

[9] 姜启源, 谢金星, 叶俊. 数学模型 [M], 北京: 高等教育出版社, 2003.

附录 A RGV 动态仿真示意图

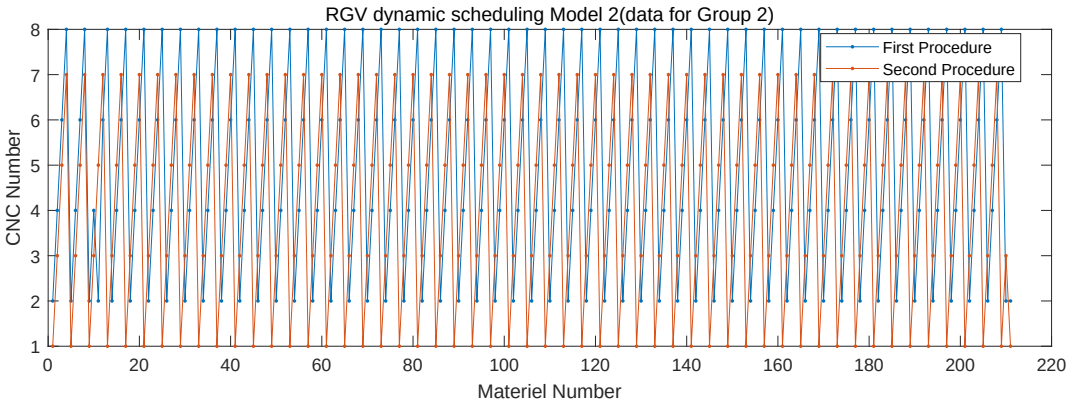

图 **A-1**　两道工序情况下 **RGV** 动态调度示意图 **(**第二组数据**)**

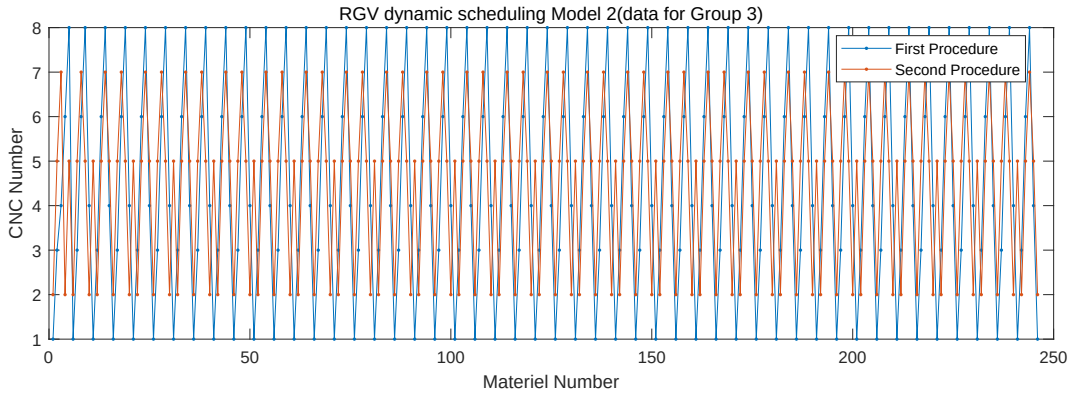

图 **A-2**　两道工序情况下 **RGV** 动态调度示意图 **(**第三组数据**)**

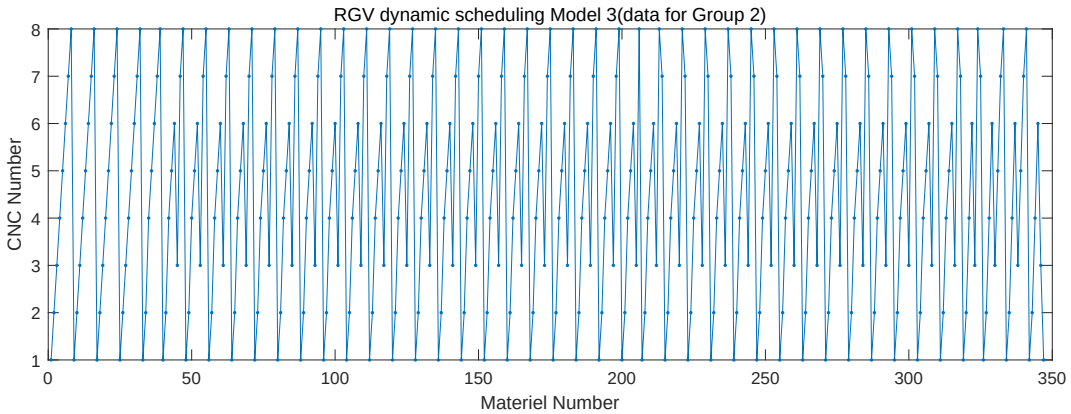

图 **A-3**　考虑故障时一道工序情况下 **RGV** 动态调度示意图 **(**第二组数据**)**

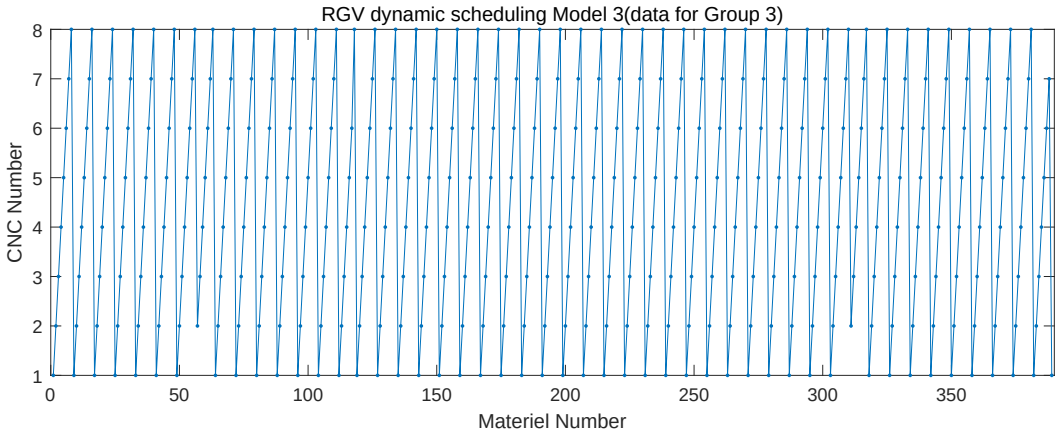

图 **A-4**　考虑故障时一道工序情况下 **RGV** 动态调度示意图 **(**第三组数据**)**

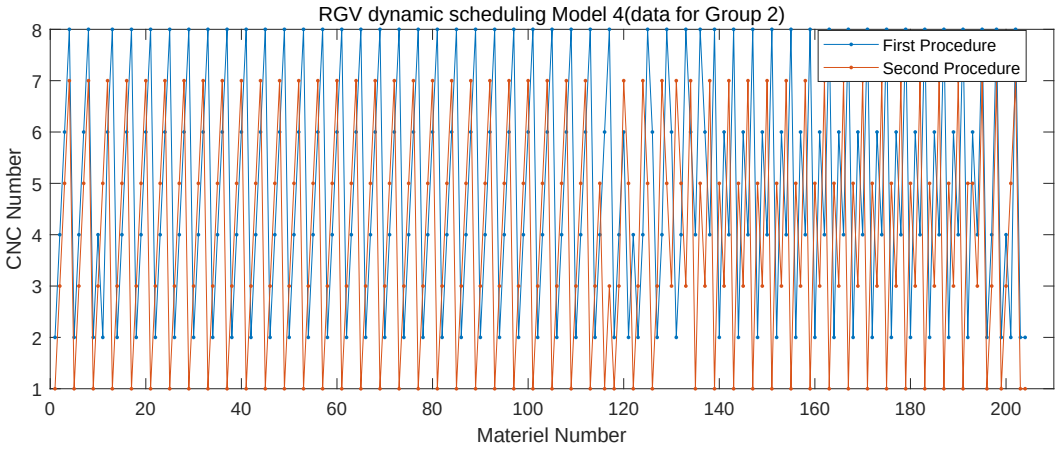

图 **A-5**　考虑故障时两道工序情况下 **RGV** 动态调度示意图 **(**第二组数据**)**

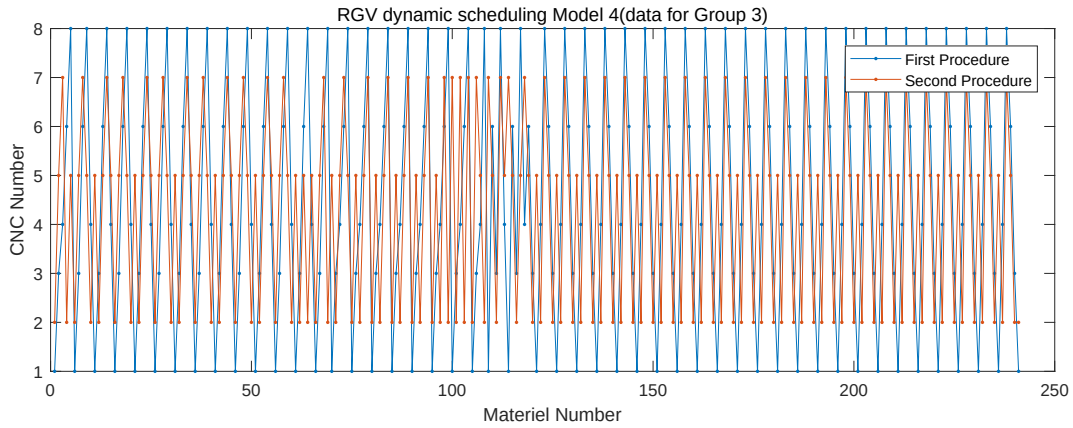

图 **A-6**　考虑故障时两道工序情况下 **RGV** 动态调度示意图 **(**第三组数据**)**

附录 B 一道工序 (无故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold
Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold
Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   初始化MATLAB
clear
clc

%   数据初始化
%   第一组数据
tm1 = 20;                   %   RGV移动1个单位时间
tm2 = 33;                   %   RGV移动2个单位时间
tm3 = 46;                   %   RGV移动3个单位时间
tcnc = 560;                 %   CNC加工完成一道工序时间
trwo = 28;                  %   RGV为奇数CNC上下料时间
trwe = 31;                  %   RGV为偶数CNC上下料时间
tclr = 25;                  %   RGV清洗熟料时间
%   第二组数据
%tm1 = 23;                   %   RGV移动1个单位时间
%tm2 = 41;                   %   RGV移动2个单位时间
%tm3 = 59;                   %   RGV移动3个单位时间
%tcnc = 580;                 %   CNC加工完成一道工序时间
%trwo = 30;                  %   RGV为奇数CNC上下料时间
%trwe = 35;                  %   RGV为偶数CNC上下料时间
%tclr = 30;                  %   RGV清洗熟料时间
%   第三组数据
%tm1 = 18;                   %   RGV移动1个单位时间
%tm2 = 32;                   %   RGV移动2个单位时间
%tm3 = 46;                   %   RGV移动3个单位时间
%tcnc = 545;                 %   CNC加工完成一道工序时间
%trwo = 27;                  %   RGV为奇数CNC上下料时间
%trwe = 32;                  %   RGV为偶数CNC上下料时间
%tclr = 25;                  %   RGV清洗熟料时间
Twork = 28800;              %   总工作时间
CNCnum = 8;                 %   CNC机器数
t = 0;                      %   时间初始化
Pos = 1;                    %   位置初始化
CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);   %   RGV移动时间矩阵
```

```
Trgvw = zeros(1, CNCnum);    %  RGV工作时间矩阵
Tcncw = zeros(1, CNCnum);    %  CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);   %  总时间矩阵
paw = 0;                     %  机械爪上是否有熟料
Tclear = 100000;             %  清洗剩余时间
tmin = 10000;                %  循环变量，表示当前步骤进行的最短时间
minPos = -1;                 %  循环变量，表示当前对哪台机器操作
count = zeros(1, CNCnum);    %  计算每台机器所上料的数目
starttime = zeros(50, CNCnum); % 每台机器上料所对应的时间
endtime = zeros(50, CNCnum); % 每台机器下料对应时间
TotalProduct = 0;            %  总的物料产量

while t < Twork
    %  根据RGV当前位置计算对应的RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
         case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
        case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
      case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
```

```
        Trgvm(5) = tm1;
        Trgvm(6) = tm1;
     Trgvm(7) = 0;
        Trgvm(8) = 0;
 end
 %   计算RGV工作时间矩阵
Trgvw(1) = trwo;
Trgvw(3) = trwo;
 Trgvw(5) = trwo;
Trgvw(7) = trwo;
Trgvw(2) = trwe;
Trgvw(4) = trwe;
Trgvw(6) = trwe;
Trgvw(8) = trwe;
 %   计算总时间
 Ttotal = Trgvm + Trgvw + Tcncw;
 %   生成最短路径位置及时间
 [tmin, minPos] = min(Ttotal);
 %   若机械爪上有熟料，则将熟料清洗时间算入
 if paw == 1
     Tclear = tclr;
 else
     Tclear = 100000;
 end
 %   若最短时间大于清洗时间，则先进行清洗
 if tmin > Tclear
     t = t + tclr;
     Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - tclr;
     paw = paw - 1;
 %   若需要操作的设备不在当前位置，则移动
 elseif ceil(minPos/2) ~= Pos
     Pos = ceil(minPos/2);
     t = t + Trgvm(minPos);
     Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvm(minPos);
 %   若需要操作的位置在当前设备
 else
     %   若当前设备未工作，进行上料操作并计数
     if CNCw(minPos) == 0
         count(minPos) = count(minPos) + 1;
         starttime(count(minPos), minPos) = t;
         t = t + Trgvw(minPos);
         Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
         Tcncw(Tcncw<0) = 0;
         Tcncw(minPos) = Tcncw(minPos) + tcnc;
         CNCw(minPos) = 1;
     %   若当前设备正在工作
     else
```

```
            %   若设备已工作完毕，进行下料操作并计数
            if Tcncw(minPos) == 0
                endtime(count(minPos), minPos) = t;
                count(minPos) = count(minPos) + 1;
                starttime(count(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                paw = paw + 1;
                Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                Tcncw(minPos) = Tcncw(minPos) + tcnc;
                CNCw(minPos) = 1;
            %   若设备正在工作，等待
            else
                t = t + 1;
                Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
            end
        end
    end
    %   去除小于0的数
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   判断每个产出物料是否有效
Logiccount = endtime > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
starttime
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
endtime
%   输出总产量
TotalProduct = sum(Logiccount(:))
```

附录 C 两道工序 (第 1 组数据，无故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   MATLAB初始化
clear
clc
```

```
4
5   %   由于第二个模型加工物料需要两道工序，因此需要考虑CNC如何进行分组，每组
6   %   分配多少个CNC的问题。在这里先进行按照（4,4）、（3,5）、（5,3）的分组结构
7   %   进行循环遍历，发现（4,4）分组的某一种方式得到的物料最多，因此给出（4,4）
8   %   分组如何遍历，以及遍历完毕后输出最优解。而（3,5）分组遍历在程序结束后的
9   %   注释给出。(5,3)分组遍历只需将相应的第一道工序和第二道工序对调即可。
10
11  %   通过计算得到，第一道工序CNC序号为1,3,5,7；第二道工序CNC序号为2,4,6，8
12  %   （最优解不唯一）
13
14  %   数据初始化
15  load('firstgroup4.mat');
16  all = [1, 2, 3, 4, 5, 6, 7, 8];
17  groupcount = zeros(70, 1);
18  secondgroup = zeros(70, 4);
19
20  %   对（4,4）分组的每种情况进行遍历
21  for divgroup = 1: 70
22      %   得到第二道工序的CNC序号
23      secondgroup(divgroup,:) = setdiff(all, firstgroup(divgroup,:));
24      tm1 = 20;                   %   RGV移动1个单位时间
25      tm2 = 33;                   %   RGV移动2个单位时间
26      tm3 = 46;                   %   RGV移动3个单位时间
27      tcnc1 = 400;                %   CNC完成第一道工序所需时间
28      tcnc2 = 378;                %   CNC完成第二道工序所需时间
29      trwo = 28;                  %   RGV为奇数CNC上下料时间
30      trwe = 31;                  %   RGV为偶数CNC上下料时间
31      tclr = 25;                  %   RGV清洗熟料时间
32      Twork = 28800;              %   总工作时间
33      CNCnum = 8;                 %   CNC机器数
34      t = 0;                      %   时间初始化
35      Pos = 1;                    %   位置初始化
36      CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
37      Trgvm = zeros(1, CNCnum);   %   RGV移动时间矩阵
38      Trgvw = zeros(1, CNCnum);   %   RGV工作时间矩阵
39      Tncw = zeros(1, CNCnum);    %   CNC工作时间矩阵
40      Ttotal = zeros(1, CNCnum);  %   总时间矩阵
41      pawsecond = 0;              %   需要进行第二道工序的物料
42      pawclear = 0;               %   需要清洗的物料
43      Tclear = 100000;            %   清洗剩余时间
44      count1 = zeros(1, CNCnum);  %   计算每台机器第一道工序上料数目
45      count2 = zeros(1, CNCnum);  %   计算每台机器第二道工序上料数目
46      starttime1 = zeros(70, CNCnum);% 每台机器第一道工序上料所对应时间
47      starttime2 = zeros(70, CNCnum);% 每台机器第二道工序上料所对应时间
48      endtime1 = zeros(70, CNCnum); % 每台机器第一道工序下料所对应时间
49      endtime2 = zeros(70, CNCnum); % 每台机器第二道工序下料所对应时间
50
```

```
%   总时间不超过8小时
while t < Twork
    %   根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
        case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
        case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
        case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = 0;
            Trgvm(8) = 0;
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
    Trgvw(3) = trwo;
    Trgvw(5) = trwo;
    Trgvw(7) = trwo;
    Trgvw(2) = trwe;
```

```
        Trgvw(4) = trwe;
        Trgvw(6) = trwe;
        Trgvw(8) = trwe;
        %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
        if pawsecond == 0
            Trgvw(secondgroup(divgroup, 1)) = ...
                Trgvw(secondgroup(divgroup, 1)) + 100000;
            Trgvw(secondgroup(divgroup, 2)) = ...
                Trgvw(secondgroup(divgroup, 2)) + 100000;
            Trgvw(secondgroup(divgroup, 3)) = ...
                Trgvw(secondgroup(divgroup, 3)) + 100000;
            Trgvw(secondgroup(divgroup, 4)) = ...
                Trgvw(secondgroup(divgroup, 4)) + 100000;
        else
            Trgvw(firstgroup(divgroup, 1)) = ...
                Trgvw(firstgroup(divgroup, 1)) + 100000;
            Trgvw(firstgroup(divgroup, 2)) = ...
                Trgvw(firstgroup(divgroup, 2)) + 100000;
            Trgvw(firstgroup(divgroup, 3)) = ...
                Trgvw(firstgroup(divgroup, 3)) + 100000;
            Trgvw(firstgroup(divgroup, 4)) = ...
                Trgvw(firstgroup(divgroup, 4)) + 100000;
        end
        %   计算总时间矩阵
        Ttotal = Trgvm + Trgvw + Tcncw;
        %   计算下一步最短时间及路径
        [tmin, minPos] = min(Ttotal);
        %   若机械爪上有待清洗的熟料，计算清洗时间
        if pawclear > 0
            Tclear = tclr;
        else
            Tclear = 100000;
        end
        %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
        if tmin > Tclear
            t = t + tclr;
            Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - tclr;
            pawclear = pawclear - 1;
        %   若下一步要操作的对象不在此处，则移动RGV
        elseif ceil(minPos/2) ~= Pos
            Pos = ceil(minPos/2);
            t = t + Trgvm(minPos);
            Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvm(minPos);
        %   若下一步要操作的对向在此处
        else
            switch minPos
                %   若操作对象为第一道工序CNC
```

```
            case {firstgroup(divgroup, 1), firstgroup(divgroup, 2), ...
                    firstgroup(divgroup, 3), firstgroup(divgroup, 4)}
                %   若操作对象未开始工作，进行上料处理并计数
                if CNCw(minPos) == 0
                    count1(minPos) = count1(minPos) + 1;
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                    CNCw(minPos) = 1;
                %   若操作对象的工作状态标记为1
                else
                    %   若操作对象已工作完毕，则先下料再上料并计数
                    if Tcncw(minPos) == 0
                        endtime1(count1(minPos), minPos) = t;
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawsecond = pawsecond + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                        CNCw(minPos) = 1;
                    %   若操作对象未工作完毕，则等待
                    else
                        t = t + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                    end
                end
            %   若操作对象为第二道工序CNC
            case {secondgroup(divgroup, 1), secondgroup(divgroup, 2), ...
                    secondgroup(divgroup, 3), secondgroup(divgroup, 4)}
                %   若操作对象未开始工作，上料并计数
                if CNCw(minPos) == 0
                    count2(minPos) = count2(minPos) + 1;
                    starttime2(count2(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                    CNCw(minPos) = 1;
                    pawsecond = pawsecond - 1;
                %   若操作对象工作状态标记为1
                else
                    %   若操作对象已工作完毕，则先下料再上料并计数
                    if Tcncw(minPos) == 0
```

```
                        endtime2(count2(minPos), minPos) = t;
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawclear = pawclear + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                        CNCw(minPos) = 1;
                        pawsecond = pawsecond - 1;
                    %   若操作对象未工作完毕，则等待
                    else
                        t = t + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                    end
                end
            end
        end
        Tcncw(Tcncw<0) = 0;
    end
    %   计算每种情况所生产的物料数
    logiccount = endtime2 > 0;
    groupcount(divgroup) = sum(logiccount(:));
end

%   找出能生产出最大物料的组合方式
[Totalcount, divgroup] = max(groupcount);
%   重新计算每个物料的第一道和第二道工序的上料和下料时间
%   得到第二道工序的CNC序号
secondgroup(divgroup,:) = setdiff(all, firstgroup(divgroup,:));
tm1 = 20;                   %   RGV移动1个单位时间
tm2 = 33;                   %   RGV移动2个单位时间
tm3 = 46;                   %   RGV移动3个单位时间
tcnc1 = 400;                %   CNC完成第一道工序所需时间
tcnc2 = 378;                %   CNC完成第二道工序所需时间
trwo = 28;                  %   RGV为奇数CNC上下料时间
trwe = 31;                  %   RGV为偶数CNC上下料时间
tclr = 25;                  %   RGV清洗熟料时间
Twork = 28800;              %   总工作时间
CNCnum = 8;                 %   CNC机器数
t = 0;                      %   时间初始化
Pos = 1;                    %   位置初始化
CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);    %   RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);   %   RGV工作时间矩阵
Tcncw = zeros(1, CNCnum);   %   CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);  %   总时间矩阵
```

```
pawsecond = 0;                %   需要进行第二道工序的物料
pawclear = 0;                 %   需要清洗的物料
Tclear = 100000;              %   清洗剩余时间
count1 = zeros(1, CNCnum);    %   计算每台机器第一道工序上料数目
count2 = zeros(1, CNCnum);    %   计算每台机器第二道工序上料数目
starttime1 = zeros(70, CNCnum);% 每台机器第一道工序上料所对应时间
starttime2 = zeros(70, CNCnum);% 每台机器第二道工序上料所对应时间
endtime1 = zeros(70, CNCnum); % 每台机器第一道工序下料所对应时间
endtime2 = zeros(70, CNCnum); % 每台机器第二道工序下料所对应时间

%   总时间不超过8小时
while t < Twork
    %   根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
        case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
        case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
        case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
```

```
        Trgvm(6) = tm1;
        Trgvm(7) = 0;
        Trgvm(8) = 0;
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
    Trgvw(3) = trwo;
    Trgvw(5) = trwo;
    Trgvw(7) = trwo;
    Trgvw(2) = trwe;
    Trgvw(4) = trwe;
    Trgvw(6) = trwe;
    Trgvw(8) = trwe;
    %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
    if pawsecond == 0
        Trgvw(secondgroup(divgroup, 1)) = Trgvw(secondgroup(divgroup, 1)) + 100000;
        Trgvw(secondgroup(divgroup, 2)) = Trgvw(secondgroup(divgroup, 2)) + 100000;
        Trgvw(secondgroup(divgroup, 3)) = Trgvw(secondgroup(divgroup, 3)) + 100000;
        Trgvw(secondgroup(divgroup, 4)) = Trgvw(secondgroup(divgroup, 4)) + 100000;
    else
        Trgvw(firstgroup(divgroup, 1)) = Trgvw(firstgroup(divgroup, 1)) + 100000;
        Trgvw(firstgroup(divgroup, 2)) = Trgvw(firstgroup(divgroup, 2)) + 100000;
        Trgvw(firstgroup(divgroup, 3)) = Trgvw(firstgroup(divgroup, 3)) + 100000;
        Trgvw(firstgroup(divgroup, 4)) = Trgvw(firstgroup(divgroup, 4)) + 100000;
    end
    %   计算总时间矩阵
    Ttotal = Trgvm + Trgvw + Tcncw;
    %   计算下一步最短时间及路径
    [tmin, minPos] = min(Ttotal);
    %   若机械爪上有待清洗的熟料，计算清洗时间
    if pawclear > 0
        Tclear = tclr;
    else
        Tclear = 100000;
    end
    %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
    if tmin > Tclear
        t = t + tclr;
        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - tclr;
        pawclear = pawclear - 1;
    %   若下一步要操作的对象不在此处，则移动RGV
    elseif ceil(minPos/2) ~= Pos
        Pos = ceil(minPos/2);
        t = t + Trgvm(minPos);
        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvm(minPos);
    %   若下一步要操作的对向在此处
    else
```

```
        switch minPos
            %   若操作对象为第一道工序CNC
            case {firstgroup(divgroup, 1), firstgroup(divgroup, 2), ...
                    firstgroup(divgroup, 3), firstgroup(divgroup, 4)}
                %   若操作对象未开始工作，进行上料处理并计数
                if CNCw(minPos) == 0
                    count1(minPos) = count1(minPos) + 1;
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                    CNCw(minPos) = 1;
                %   若操作对象的工作状态标记为1
                else
                    %   若操作对象已工作完毕，则先下料再上料并计数
                    if Tcncw(minPos) == 0
                        endtime1(count1(minPos), minPos) = t;
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawsecond = pawsecond + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                        CNCw(minPos) = 1;
                    %   若操作对象未工作完毕，则等待
                    else
                        t = t + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                    end
                end
            %   若操作对象为第二道工序CNC
            case {secondgroup(divgroup, 1), secondgroup(divgroup, 2), ...
                    secondgroup(divgroup, 3), secondgroup(divgroup, 4)}
                %   若操作对象未开始工作，上料并计数
                if CNCw(minPos) == 0
                    count2(minPos) = count2(minPos) + 1;
                    starttime2(count2(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                    CNCw(minPos) = 1;
                    pawsecond = pawsecond - 1;
                %   若操作对象工作状态标记为1
                else
```

```
                    %   若操作对象已工作完毕，则先下料再上料并计数
                    if Tcncw(minPos) == 0
                        endtime2(count2(minPos), minPos) = t;
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawclear = pawclear + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                        CNCw(minPos) = 1;
                        pawsecond = pawsecond - 1;
                    %   若操作对象未工作完毕，则等待
                    else
                        t = t + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                    end
                end
            end
        end
        Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   判断每个产出物料是否有效
logiccount = endtime2 > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
%   分第一道工序和第二道工序
starttime1
starttime2
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   分第一道工序和第二道工序
endtime1
endtime2
%   输出总产量
TotalProduct = sum(logiccount(:))



% %   若为（3,5）分组，遍历代码
```

```
% %  MATLAB初始化
% clear
% clc
%
% %  数据初始化
% load('firstgroup3.mat');
% all = [1, 2, 3, 4, 5, 6, 7, 8];
% groupcount = zeros(70, 1);
% secondgroup = zeros(70, 5);
%
%
% for divgroup = 1: 56
%     secondgroup(divgroup,:) = setdiff(all, firstgroup3(divgroup,:));
%     tm1 = 20;                    %  RGV移动1个单位时间
%     tm2 = 33;                    %  RGV移动2个单位时间
%     tm3 = 46;                    %  RGV移动3个单位时间
%     tcnc1 = 400;                 %  CNC完成第一道工序所需时间
%     tcnc2 = 378;                 %  CNC完成第二道工序所需时间
%     trwo = 28;                   %  RGV为奇数CNC上下料时间
%     trwe = 31;                   %  RGV为偶数CNC上下料时间
%     tclr = 25;                   %  RGV清洗熟料时间
%     Twork = 28800;               %  总工作时间
%     CNCnum = 8;                  %  CNC机器数
%     t = 0;                       %  时间初始化
%     Pos = 1;                     %  位置初始化
%     CNCw = zeros(1, CNCnum);     %  CNC工作状态标志
%     Trgvm = zeros(1, CNCnum);     %  RGV移动时间矩阵
%     Trgvw = zeros(1, CNCnum);    %  RGV工作时间矩阵
%     Tncw = zeros(1, CNCnum);     %  CNC工作时间矩阵
%     Ttotal = zeros(1, CNCnum);   %  总时间矩阵
%     pawsecond = 0;               %  需要进行第二道工序的物料
%     pawclear = 0;                %  需要清洗的物料
%     Tclear = 100000;             %  清洗剩余时间
%     count1 = zeros(1, CNCnum);   %  计算每台机器第一道工序上料数目
%     count2 = zeros(1, CNCnum);   %  计算每台机器第二道工序上料数目
%     starttime1 = zeros(100, CNCnum);% 每台机器第一道工序上料所对应时间
%     starttime2 = zeros(100, CNCnum);% 每台机器第二道工序上料所对应时间
%     endtime1 = zeros(100, CNCnum); % 每台机器第一道工序下料所对应时间
%     endtime2 = zeros(100, CNCnum); % 每台机器第二道工序下料所对应时间
%     %  总时间不超过8小时
%     while t < Twork
%         %  根据RGV所在位置计算RGV移动时间矩阵
%         switch Pos
%             case 1
%                 Trgvm(1) = 0;
%                 Trgvm(2) = 0;
%                 Trgvm(3) = tm1;
```

```
474 %                 Trgvm(4) = tm1;
475 %                 Trgvm(5) = tm2;
476 %                 Trgvm(6) = tm2;
477 %                 Trgvm(7) = tm3;
478 %                 Trgvm(8) = tm3;
479 %             case 2
480 %                 Trgvm(1) = tm1;
481 %                 Trgvm(2) = tm1;
482 %                 Trgvm(3) = 0;
483 %                 Trgvm(4) = 0;
484 %                 Trgvm(5) = tm1;
485 %                 Trgvm(6) = tm1;
486 %                 Trgvm(7) = tm2;
487 %                 Trgvm(8) = tm2;
488 %             case 3
489 %                 Trgvm(1) = tm2;
490 %                 Trgvm(2) = tm2;
491 %                 Trgvm(3) = tm1;
492 %                 Trgvm(4) = tm1;
493 %                 Trgvm(5) = 0;
494 %                 Trgvm(6) = 0;
495 %                 Trgvm(7) = tm1;
496 %                 Trgvm(8) = tm1;
497 %             case 4
498 %                 Trgvm(1) = tm3;
499 %                 Trgvm(2) = tm3;
500 %                 Trgvm(3) = tm2;
501 %                 Trgvm(4) = tm2;
502 %                 Trgvm(5) = tm1;
503 %                 Trgvm(6) = tm1;
504 %                 Trgvm(7) = 0;
505 %                 Trgvm(8) = 0;
506 %         end
507 %         %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
508 %         Trgvw(1) = trwo;
509 %         Trgvw(3) = trwo;
510 %         Trgvw(5) = trwo;
511 %         Trgvw(7) = trwo;
512 %         Trgvw(2) = trwe;
513 %         Trgvw(4) = trwe;
514 %         Trgvw(6) = trwe;
515 %         Trgvw(8) = trwe;
516 %         %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
517 %         if pawsecond == 0
518 %             Trgvw(secondgroup(divgroup, 1)) = Trgvw(secondgroup(divgroup, 1)) + 100000;
519 %             Trgvw(secondgroup(divgroup, 2)) = Trgvw(secondgroup(divgroup, 2)) + 100000;
520 %             Trgvw(secondgroup(divgroup, 3)) = Trgvw(secondgroup(divgroup, 3)) + 100000;
```

```
%             Trgvw(secondgroup(divgroup, 4)) = Trgvw(secondgroup(divgroup, 4)) + 100000;
%             Trgvw(secondgroup(divgroup, 5)) = Trgvw(secondgroup(divgroup, 5)) + 100000;
%         else
%             Trgvw(firstgroup3(divgroup, 1)) = Trgvw(firstgroup3(divgroup, 1)) + 100000;
%             Trgvw(firstgroup3(divgroup, 2)) = Trgvw(firstgroup3(divgroup, 2)) + 100000;
%             Trgvw(firstgroup3(divgroup, 3)) = Trgvw(firstgroup3(divgroup, 3)) + 100000;
%         end
%         %   计算总时间矩阵
%         Ttotal = Trgvm + Trgvw + Tcncw;
%         %   计算下一步最短时间及路径
%         [tmin, minPos] = min(Ttotal);
%         %   若机械爪上有待清洗的熟料，计算清洗时间
%         if pawclear > 0
%             Tclear = tclr;
%         else
%             Tclear = 100000;
%         end
%         %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
%         if tmin > Tclear
%             t = t + tclr;
%             Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - tclr;
%             pawclear = pawclear - 1;
%         %   若下一步要操作的对象不在此处，则移动RGV
%         elseif ceil(minPos/2) ~= Pos
%             Pos = ceil(minPos/2);
%             t = t + Trgvm(minPos);
%             Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvm(minPos);
%         %   若下一步要操作的对向在此处
%         else
%             switch minPos
%                 %   若操作对象为第一道工序CNC
%                 case {firstgroup3(divgroup, 1), firstgroup3(divgroup, 2), ...
%                         firstgroup3(divgroup, 3)}
%                     %   若操作对象未开始工作，进行上料处理并计数
%                     if CNCw(minPos) == 0
%                         count1(minPos) = count1(minPos) + 1;
%                         starttime1(count1(minPos), minPos) = t;
%                         t = t + Trgvw(minPos);
%                         Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
%                         Tcncw(Tcncw<0) = 0;
%                         Tcncw(minPos) = Tcncw(minPos) + tcnc1;
%                         CNCw(minPos) = 1;
%                     %   若操作对象的工作状态标记为1
%                     else
%                         %   若操作对象已工作完毕，则先下料再上料并计数
%                         if Tcncw(minPos) == 0
%                             endtime1(count1(minPos), minPos) = t;
```

```
%                         count1(minPos) = count1(minPos) + 1;
%                         starttime1(count1(minPos), minPos) = t;
%                         t = t + Trgvw(minPos);
%                         pawsecond = pawsecond + 1;
%                         Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
%                         Tcncw(Tcncw<0) = 0;
%                         Tcncw(minPos) = Tcncw(minPos) + tcnc1;
%                         CNCw(minPos) = 1;
%                     %   若操作对象未工作完毕，则等待
%                     else
%                         t = t + 1;
%                         Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
%                     end
%                 end
%             %   若操作对象为第二道工序CNC
%             case {secondgroup(divgroup, 1), secondgroup(divgroup, 2), ...
%                     secondgroup(divgroup, 3), secondgroup(divgroup, 4), ...
%                     secondgroup(divgroup, 5)}
%                 %   若操作对象未开始工作，上料并计数
%                 if CNCw(minPos) == 0
%                     count2(minPos) = count2(minPos) + 1;
%                     starttime2(count2(minPos), minPos) = t;
%                     t = t + Trgvw(minPos);
%                     Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
%                     Tcncw(Tcncw<0) = 0;
%                     Tcncw(minPos) = Tcncw(minPos) + tcnc2;
%                     CNCw(minPos) = 1;
%                     pawsecond = pawsecond - 1;
%                 %   若操作对象工作状态标记为1
%                 else
%                     %   若操作对象已工作完毕，则先下料再上料并计数
%                     if Tcncw(minPos) == 0
%                         endtime2(count2(minPos), minPos) = t;
%                         count2(minPos) = count2(minPos) + 1;
%                         starttime2(count2(minPos), minPos) = t;
%                         t = t + Trgvw(minPos);
%                         pawclear = pawclear + 1;
%                         Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
%                         Tcncw(Tcncw<0) = 0;
%                         Tcncw(minPos) = Tcncw(minPos) + tcnc2;
%                         CNCw(minPos) = 1;
%                         pawsecond = pawsecond - 1;
%                     %   若操作对象未工作完毕，则等待
%                     else
%                         t = t + 1;
%                         Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
%                     end
```

```
%                     end
%                 end
%             end
%             Tcncw(Tcncw<0) = 0;
%         end
%         logiccount = endtime2 > 0;
%         groupcount(divgroup) = sum(logiccount(:));
% end
% %   找出能生产出最大物料的组合方式
% [Totalcount, divgroup] = max(groupcount);
```

附录 D 两道工序 (第 2 组数据，无故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   MATLAB初始化
clear
clc

%   由于第二个模型加工物料需要两道工序，因此需要考虑CNC如何进行分组，每组
%   分配多少个CNC的问题。在这里先进行按照（4,4）、（3,5）、（5,3）的分组结构
%   进行循环遍历，发现（4,4）分组的某一种方式得到的物料最多，因此给出（4,4）
%   分组如何遍历，以及遍历完毕后输出最优解。而（3,5）分组和（5,3）分组遍历
%   参见CUMCM2018BModel2_1.m

%   通过计算得到，第一道工序CNC序号为2,4,6,8；第二道工序CNC序号为1,3,5,7
%   （最优解唯一）

%   数据初始化
load('firstgroup4.mat');
all = [1, 2, 3, 4, 5, 6, 7, 8];
groupcount = zeros(70, 1);
secondgroup = zeros(70, 4);

%   对（4,4）分组的每种情况进行遍历
%   相应的遍历操作不再赘述，可参见CUMCM2018BModel2_1.m
%   此处直接给出遍历得到的最优解
divgroup = 50;

%   得到第二道工序的CNC序号
secondgroup(divgroup,:) = setdiff(all, firstgroup(divgroup,:));
tm1 = 23;                   %   RGV移动1个单位时间
tm2 = 41;                   %   RGV移动2个单位时间
tm3 = 59;                   %   RGV移动3个单位时间
```

```
tcnc1 = 280;                %   CNC完成第一道工序所需时间
tcnc2 = 500;                %   CNC完成第二道工序所需时间
trwo = 30;                  %   RGV为奇数CNC上下料时间
trwe = 35;                  %   RGV为偶数CNC上下料时间
tclr = 30;                  %   RGV清洗熟料时间
Twork = 28800;              %   总工作时间
CNCnum = 8;                 %   CNC机器数
t = 0;                      %   时间初始化
Pos = 1;                    %   位置初始化
CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);     %   RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);   %   RGV工作时间矩阵
Tncw = zeros(1, CNCnum);    %   CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);  %   总时间矩阵
pawsecond = 0;              %   需要进行第二道工序的物料
pawclear = 0;               %   需要清洗的物料
Tclear = 100000;            %   清洗剩余时间
count1 = zeros(1, CNCnum);  %   计算每台机器第一道工序上料数目
count2 = zeros(1, CNCnum);  %   计算每台机器第二道工序上料数目
starttime1 = zeros(60, CNCnum);% 每台机器第一道工序上料所对应时间
starttime2 = zeros(60, CNCnum);% 每台机器第二道工序上料所对应时间
endtime1 = zeros(60, CNCnum); % 每台机器第一道工序下料所对应时间
endtime2 = zeros(60, CNCnum); % 每台机器第二道工序下料所对应时间

%   总时间不超过8小时
while t < Twork
    %   根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
        case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
        case 3
```

```
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
        case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = 0;
            Trgvm(8) = 0;
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
    Trgvw(3) = trwo;
    Trgvw(5) = trwo;
    Trgvw(7) = trwo;
    Trgvw(2) = trwe;
    Trgvw(4) = trwe;
    Trgvw(6) = trwe;
    Trgvw(8) = trwe;
    %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
    if pawsecond == 0
        Trgvw(secondgroup(divgroup, 1)) = Trgvw(secondgroup(divgroup, 1)) + 100000;
        Trgvw(secondgroup(divgroup, 2)) = Trgvw(secondgroup(divgroup, 2)) + 100000;
        Trgvw(secondgroup(divgroup, 3)) = Trgvw(secondgroup(divgroup, 3)) + 100000;
        Trgvw(secondgroup(divgroup, 4)) = Trgvw(secondgroup(divgroup, 4)) + 100000;
    else
        Trgvw(firstgroup(divgroup, 1)) = Trgvw(firstgroup(divgroup, 1)) + 100000;
        Trgvw(firstgroup(divgroup, 2)) = Trgvw(firstgroup(divgroup, 2)) + 100000;
        Trgvw(firstgroup(divgroup, 3)) = Trgvw(firstgroup(divgroup, 3)) + 100000;
        Trgvw(firstgroup(divgroup, 4)) = Trgvw(firstgroup(divgroup, 4)) + 100000;
    end
    %   计算总时间矩阵
    Ttotal = Trgvm + Trgvw + Tcncw;
    %   计算下一步最短时间及路径
    [tmin, minPos] = min(Ttotal);
    %   若机械爪上有待清洗的熟料，计算清洗时间
    if pawclear > 0
        Tclear = tclr;
    else
```

```
        Tclear = 100000;
    end
    %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
    if tmin > Tclear
        t = t + tclr;
        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - tclr;
        pawclear = pawclear - 1;
    %   若下一步要操作的对象不在此处，则移动RGV
    elseif ceil(minPos/2) ~= Pos
        Pos = ceil(minPos/2);
        t = t + Trgvm(minPos);
        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvm(minPos);
    %   若下一步要操作的对向在此处
    else
        switch minPos
            %   若操作对象为第一道工序CNC
            case {firstgroup(divgroup, 1), firstgroup(divgroup, 2), ...
                    firstgroup(divgroup, 3), firstgroup(divgroup, 4)}
                %   若操作对象未开始工作，进行上料处理并计数
                if CNCw(minPos) == 0
                    count1(minPos) = count1(minPos) + 1;
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                    CNCw(minPos) = 1;
                %   若操作对象的工作状态标记为1
                else
                    %   若操作对象已工作完毕，则先下料再上料并计数
                    if Tcncw(minPos) == 0
                        endtime1(count1(minPos), minPos) = t;
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawsecond = pawsecond + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                        CNCw(minPos) = 1;
                    %   若操作对象未工作完毕，则等待
                    else
                        t = t + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                    end
                end
            %   若操作对象为第二道工序CNC
```

```
            case {secondgroup(divgroup, 1), secondgroup(divgroup, 2), ...
                    secondgroup(divgroup, 3), secondgroup(divgroup, 4)}
                %   若操作对象未开始工作，上料并计数
                if CNCw(minPos) == 0
                    count2(minPos) = count2(minPos) + 1;
                    starttime2(count2(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                    CNCw(minPos) = 1;
                    pawsecond = pawsecond - 1;
                %   若操作对象工作状态标记为1
                else
                    %   若操作对象已工作完毕，则先下料再上料并计数
                    if Tcncw(minPos) == 0
                        endtime2(count2(minPos), minPos) = t;
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawclear = pawclear + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                        CNCw(minPos) = 1;
                        pawsecond = pawsecond - 1;
                    %   若操作对象未工作完毕，则等待
                    else
                        t = t + 1;
                        Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                    end
                end
        end
    end
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   判断每个产出物料是否有效
logiccount = endtime2 > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
%   分第一道工序和第二道工序
```

```
starttime1
starttime2
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   分第一道工序和第二道工序
endtime1
endtime2
%   输出总产量
TotalProduct = sum(logiccount(:))
```

附录 E 两道工序 (第 3 组数据，无故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   MATLAB初始化
clear
clc

%   由于第二个模型加工物料需要两道工序，因此需要考虑CNC如何进行分组，每组
%   分配多少个CNC的问题。在这里先进行按照（4,4）、（3,5）、（5,3）的分组结构
%   进行循环遍历，发现（5,3）分组的某一种方式得到的物料最多，因此给出（5,3）
%   分组如何遍历，以及遍历完毕后输出最优解。（4,4）分组遍历可参考（5,3）分组
%   遍历对相应位置进行改进，此处不再给出。
%   （5,3）分组遍历参见CUMCM2018BModel2_1.m

%   通过计算得到，第一道工序CNC序号为1,3,4,6,8；第二道工序CNC序号为2,5,7
%   （最优解不唯一）

%   数据初始化
load('firstgroup3.mat');
all = [1, 2, 3, 4, 5, 6, 7, 8];
groupcount = zeros(70, 1);
secondgroup = firstgroup3;
firstgroup = zeros(70,5);

%   对（5,3）分组的每种情况进行遍历
%   相应的遍历操作不再赘述，可参见CUMCM2018BModel2_1.m
%   此处直接给出遍历得到的最优解
divgroup = 32;
%   得到第一道工序的CNC序号
firstgroup(divgroup,:) = setdiff(all, firstgroup3(divgroup,:));
tm1 = 18;                   %   RGV移动1个单位时间
tm2 = 32;                   %   RGV移动2个单位时间
tm3 = 46;                   %   RGV移动3个单位时间
tcnc1 = 455;                %   CNC完成第一道工序所需时间
```

```
tcnc2 = 182;                %   CNC完成第二道工序所需时间
trwo = 27;                  %   RGV为奇数CNC上下料时间
trwe = 32;                  %   RGV为偶数CNC上下料时间
tclr = 25;                  %   RGV清洗熟料时间
Twork = 28800;              %   总工作时间
CNCnum = 8;                 %   CNC机器数
t = 0;                      %   时间初始化
Pos = 1;                    %   位置初始化
CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);     %   RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);   %   RGV工作时间矩阵
Tncw = zeros(1, CNCnum);    %   CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);  %   总时间矩阵
pawsecond = 0;              %   需要进行第二道工序的物料
pawclear = 0;               %   需要清洗的物料
Tclear = 100000;            %   清洗剩余时间
count1 = zeros(1, CNCnum);  %   计算每台机器第一道工序上料数目
count2 = zeros(1, CNCnum);  %   计算每台机器第二道工序上料数目
starttime1 = zeros(100, CNCnum);% 每台机器第一道工序上料所对应时间
starttime2 = zeros(100, CNCnum);% 每台机器第二道工序上料所对应时间
endtime1 = zeros(100, CNCnum); % 每台机器第一道工序下料所对应时间
endtime2 = zeros(100, CNCnum); % 每台机器第二道工序下料所对应时间
%   总时间不超过8小时
while t < Twork
    %   根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
        case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
        case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
```

```
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
        case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = 0;
            Trgvm(8) = 0;
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
    Trgvw(3) = trwo;
    Trgvw(5) = trwo;
    Trgvw(7) = trwo;
    Trgvw(2) = trwe;
    Trgvw(4) = trwe;
    Trgvw(6) = trwe;
    Trgvw(8) = trwe;
    %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
    if pawsecond == 0
        Trgvw(secondgroup(divgroup, 1)) = ...
            Trgvw(secondgroup(divgroup, 1)) + 100000;
        Trgvw(secondgroup(divgroup, 2)) = ...
            Trgvw(secondgroup(divgroup, 2)) + 100000;
        Trgvw(secondgroup(divgroup, 3)) = ...
            Trgvw(secondgroup(divgroup, 3)) + 100000;
    else
        Trgvw(firstgroup(divgroup, 1)) = ...
            Trgvw(firstgroup(divgroup, 1)) + 100000;
        Trgvw(firstgroup(divgroup, 2)) = ...
            Trgvw(firstgroup(divgroup, 2)) + 100000;
        Trgvw(firstgroup(divgroup, 3)) = ...
            Trgvw(firstgroup(divgroup, 3)) + 100000;
        Trgvw(firstgroup(divgroup, 4)) = ...
            Trgvw(firstgroup(divgroup, 1)) + 100000;
        Trgvw(firstgroup(divgroup, 5)) = ...
            Trgvw(firstgroup(divgroup, 2)) + 100000;
    end
    %   计算总时间矩阵
    Ttotal = Trgvm + Trgvw + Tcncw;
```

```
%   计算下一步最短时间及路径
rannum1 = rand(1);
if rannum1 > 0
    [tmin, minPos] = min(Ttotal);
else
    [sortTtotal, sortix] = sort(Ttotal);
    tmin = Ttotal(sortix(2));
    minPos = sortix(2);
end
%   若机械爪上有待清洗的熟料，计算清洗时间
if pawclear > 0
    Tclear = tclr;
else
    Tclear = 100000;
end
%   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
if tmin > Tclear
    t = t + tclr;
    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - tclr;
    pawclear = pawclear - 1;
%   若下一步要操作的对象不在此处，则移动RGV
elseif ceil(minPos/2) ~= Pos
    Pos = ceil(minPos/2);
    t = t + Trgvm(minPos);
    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvm(minPos);
%   若下一步要操作的对向在此处
else
    switch minPos
        %   若操作对象为第一道工序CNC
        case {firstgroup(divgroup, 1), firstgroup(divgroup, 2), ...
              firstgroup(divgroup, 3), firstgroup(divgroup, 4), ...
              firstgroup(divgroup, 5)}
            %   若操作对象未开始工作，进行上料处理并计数
            if CNCw(minPos) == 0
                count1(minPos) = count1(minPos) + 1;
                starttime1(count1(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                CNCw(minPos) = 1;
            %   若操作对象的工作状态标记为1
            else
                %   若操作对象已工作完毕，则先下料再上料并计数
                if Tcncw(minPos) == 0
                    endtime1(count1(minPos), minPos) = t;
                    count1(minPos) = count1(minPos) + 1;
```

```
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    pawsecond = pawsecond + 1;
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                    CNCw(minPos) = 1;
                %   若操作对象未工作完毕，则等待
                else
                    t = t + 1;
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                end
            end
        %   若操作对象为第二道工序CNC
        case {secondgroup(divgroup, 1), secondgroup(divgroup, 2), ...
                secondgroup(divgroup, 3)}
            %   若操作对象未开始工作，上料并计数
            if CNCw(minPos) == 0
                count2(minPos) = count2(minPos) + 1;
                starttime2(count2(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                CNCw(minPos) = 1;
                pawsecond = pawsecond - 1;
            %   若操作对象工作状态标记为1
            else
                %   若操作对象已工作完毕，则先下料再上料并计数
                if Tcncw(minPos) == 0
                    endtime2(count2(minPos), minPos) = t;
                    count2(minPos) = count2(minPos) + 1;
                    starttime2(count2(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    pawclear = pawclear + 1;
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                    CNCw(minPos) = 1;
                    pawsecond = pawsecond - 1;
                %   若操作对象未工作完毕，则等待
                else
                    t = t + 1;
                    Tcncw(CNCw == 1) = Tcncw(CNCw == 1) - 1;
                end
            end
    end
```

```
    end
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   判断每个产出物料是否有效
logiccount = endtime2 > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
%   分第一道工序和第二道工序
starttime1
starttime2
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   分第一道工序和第二道工序
endtime1
endtime2
%   输出总产量
TotalProduct = sum(logiccount(:))
```

附录 F 一道工序 (有故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   初始化MATLAB
clear
clc

%   数据初始化
tm1 = 20;                   %   RGV移动1个单位时间
tm2 = 33;                   %   RGV移动2个单位时间
tm3 = 46;                   %   RGV移动3个单位时间
tcnc = 560;                 %   CNC加工完成一道工序时间
trwo = 28;                  %   RGV为奇数CNC上下料时间
trwe = 31;                  %   RGV为偶数CNC上下料时间
tclr = 25;                  %   RGV清洗熟料时间
%   第二组数据
%tm1 = 23;                   %   RGV移动1个单位时间
%tm2 = 41;                   %   RGV移动2个单位时间
%tm3 = 59;                   %   RGV移动3个单位时间
%tcnc = 580;                 %   CNC加工完成一道工序时间
```

```
%trwo = 30;                    %   RGV为奇数CNC上下料时间
%trwe = 35;                    %   RGV为偶数CNC上下料时间
%tclr = 30;                    %   RGV清洗熟料时间
%   第三组数据
%tm1 = 18;                     %   RGV移动1个单位时间
%tm2 = 32;                     %   RGV移动2个单位时间
%tm3 = 46;                     %   RGV移动3个单位时间
%tcnc = 545;                   %   CNC加工完成一道工序时间
%trwo = 27;                    %   RGV为奇数CNC上下料时间
%trwe = 32;                    %   RGV为偶数CNC上下料时间
%tclr = 25;                    %   RGV清洗熟料时间
Twork = 28800;                 %   总工作时间
CNCnum = 8;                    %   CNC机器数
t = 0;                         %   时间初始化
Pos = 1;                       %   位置初始化
CNCw = zeros(1, CNCnum);       %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);      %   RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);      %   RGV工作时间矩阵
Tcncw = zeros(1, CNCnum);      %   CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);     %   总时间矩阵
paw = 0;                       %   机械爪上是否有熟料
Tclear = 100000;               %   清洗剩余时间
tmin = 10000;                  %   循环变量，表示当前步骤进行的最短时间
minPos = -1;                   %   循环变量，表示当前对哪台机器操作
count = zeros(1, CNCnum);      %   计算每台机器所上料的数目
starttime = zeros(50, CNCnum); % 每台机器上料所对应的时间
endtime = zeros(50, CNCnum); % 每台机器下料对应时间
sortTtotal = zeros(1, CNCnum); % 模拟退火算法所需的排序矩阵
sortix = zeros(1, CNCnum);     %   模拟退火算法所需的位置矩阵
rannum1 = 1;                   %   随机数1，用于模拟退火时的计算
rannum2 = 1;                   %   随机数2，用于对CNC机器的故障判定
rannum3 = 1;                   %   随机数3，用于确定故障发生的时间
Accrecord = zeros(10, 3);      %   故障记录矩阵
Acccount = 0;                  %   故障记录

while t < Twork
    %   根据RGV当前位置计算对应的RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
```

```
        case 2
           Trgvm(1) = tm1;
           Trgvm(2) = tm1;
           Trgvm(3) = 0;
           Trgvm(4) = 0;
           Trgvm(5) = tm1;
           Trgvm(6) = tm1;
           Trgvm(7) = tm2;
           Trgvm(8) = tm2;
       case 3
           Trgvm(1) = tm2;
           Trgvm(2) = tm2;
           Trgvm(3) = tm1;
           Trgvm(4) = tm1;
           Trgvm(5) = 0;
           Trgvm(6) = 0;
           Trgvm(7) = tm1;
           Trgvm(8) = tm1;
     case 4
           Trgvm(1) = tm3;
           Trgvm(2) = tm3;
           Trgvm(3) = tm2;
           Trgvm(4) = tm2;
           Trgvm(5) = tm1;
           Trgvm(6) = tm1;
        Trgvm(7) = 0;
           Trgvm(8) = 0;
    end
    %   计算RGV工作时间矩阵
   Trgvw(1) = trwo;
   Trgvw(3) = trwo;
    Trgvw(5) = trwo;
   Trgvw(7) = trwo;
   Trgvw(2) = trwe;
   Trgvw(4) = trwe;
   Trgvw(6) = trwe;
   Trgvw(8) = trwe;
    %   计算总时间
    Ttotal = Trgvm + Trgvw + Tcncw;
    %   找出最短路径
    %   基于模拟退火算法生成最短路径位置
    rannum1 = rand(1);
    if rannum1 > 0.02
        [tmin, minPos] = min(Ttotal);
    else
        [sortTtotal, sortix] = sort(Ttotal);
        tmin = Ttotal(sortix(2));
```

```
        minPos = sortix(2);
    end
    %   若机械爪上有熟料，则与最短路径相比较
    if paw == 1
        Tclear = tclr;
    else
        Tclear = 100000;
    end
    %   若最短时间大于清洗时间，则先进行清洗
    if tmin > Tclear
        t = t + tclr;
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - tclr;
        paw = paw - 1;
    %   若需要操作的设备不在当前位置，则移动
    elseif ceil(minPos/2) ~= Pos
        Pos = ceil(minPos/2);
        t = t + Trgvm(minPos);
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvm(minPos);
    %   若需要操作的位置在当前设备
    else
        %   若当前设备未工作，进行上料操作并计数
        if CNCw(minPos) == 0
            count(minPos) = count(minPos) + 1;
            starttime(count(minPos), minPos) = t;
            t = t + Trgvw(minPos);
            Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
            Tcncw(Tcncw<0) = 0;
            %   根据概率计算机器是否发生故障，并生成故障发生时间以及人工修复时间
            rannum2 = rand(1);
            if rannum2 > 0.01
                Tcncw(minPos) = Tcncw(minPos) + tcnc;
                CNCw(minPos) = 1;
            else
                rannum3 = ceil(rand(1) * tcnc);
                Acccount = Acccount + 1;
                Accrecord(Acccount, 1) = minPos;
                Accrecord(Acccount, 2) = t + rannum3;
                Accrecord(Acccount, 3) = t + rannum3 + ceil((10+1000*rannum2)*60);
                CNCw(minPos) = 2;
                Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
            end
        %   若当前设备正在工作
        else
            %   若设备已工作完毕，进行下料操作并计数
            if Tcncw(minPos) == 0
                if CNCw(minPos) == 1
                    endtime(count(minPos), minPos) = t;
```

```
                count(minPos) = count(minPos) + 1;
                starttime(count(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                paw = paw + 1;
                Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                %   根据概率计算机器是否发生故障，并生成故障发生时间以及人工修复时间
                rannum2 = rand(1);
                if rannum2 > 0.01
                    Tcncw(minPos) = Tcncw(minPos) + tcnc;
                    CNCw(minPos) = 1;
                else
                    rannum3 = ceil(rand(1) * tcnc);
                    Acccount = Acccount + 1;
                    Accrecord(Acccount, 1) = minPos;
                    Accrecord(Acccount, 2) = t + rannum3;
                    Accrecord(Acccount, 3) = t + rannum3 + ceil((10+1000*rannum2)*60);
                    CNCw(minPos) = 2;
                    Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
                end
            else
                count(minPos) = count(minPos) + 1;
                starttime(count(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                %   根据概率计算机器是否发生故障，并生成故障发生时间以及人工修复时间
                rannum2 = rand(1);
                if rannum2 > 0.01
                    Tcncw(minPos) = Tcncw(minPos) + tcnc;
                    CNCw(minPos) = 1;
                else
                    rannum3 = ceil(rand(1) * tcnc);
                    Acccount = Acccount + 1;
                    Accrecord(Acccount, 1) = minPos;
                    Accrecord(Acccount, 2) = t + rannum3;
                    Accrecord(Acccount, 3) = t + rannum3 + ceil((10+1000*rannum2)*60);
                    CNCw(minPos) = 2;
                    Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
                end
            end
        %   若设备正在工作，等待
        else
            t = t + 1;
            Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
        end
    end
```

```
    end
    %   去除小于0的数
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   除此之外，由于该模型在寻找最短路径时采用了模拟退火算法，因此得到的结果
%   与Excel可能有一定出入

%   由于机器的损坏也具有一定概率，所以得到的结果与Excel可能有一定出入

%   判断每个产出物料是否有效
Logiccount = endtime > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
starttime
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   endtime为0说明该物料生产过程中出现故障
endtime
%   输出故障记录
Accrecord
%   输出总产量
TotalProduct = sum(Logiccount(:))
```

附录 G 两道工序 (第 1 组数据，有故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   MATLAB初始化
clear
clc

%   数据初始化
tm1 = 20;                   %   RGV移动1个单位时间
tm2 = 33;                   %   RGV移动2个单位时间
tm3 = 46;                   %   RGV移动3个单位时间
tcnc1 = 400;                %   CNC完成第一道工序所需时间
tcnc2 = 378;                %   CNC完成第二道工序所需时间
trwo = 28;                  %   RGV为奇数CNC上下料时间
trwe = 31;                  %   RGV为偶数CNC上下料时间
```

```
tclr = 25;                  %   RGV清洗熟料时间
Twork = 28800;              %   总工作时间
CNCnum = 8;                 %   CNC机器数
t = 0;                      %   时间初始化
Pos = 1;                    %   位置初始化
CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);    %   RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);   %   RGV工作时间矩阵
Tcncw = zeros(1, CNCnum);   %   CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);  %   总时间矩阵
pawsecond = 0;              %   需要进行第二道工序的物料
pawclear = 0;               %   需要清洗的物料
Tclear = 100000;            %   清洗剩余时间
count1 = zeros(1, CNCnum);  %   计算每台机器第一道工序上料数目
count2 = zeros(1, CNCnum);  %   计算每台机器第二道工序上料数目
starttime1 = zeros(70, CNCnum);% 每台机器第一道工序上料所对应时间
starttime2 = zeros(70, CNCnum);% 每台机器第二道工序上料所对应时间
endtime1 = zeros(70, CNCnum); % 每台机器第一道工序下料所对应时间
endtime2 = zeros(70, CNCnum); % 每台机器第二道工序下料所对应时间
sortTtotal = zeros(1, CNCnum); % 模拟退火算法所需的排序矩阵
sortix = zeros(1, CNCnum);  %   模拟退火算法所需的位置矩阵
rannum1 = 1;                %   随机数1，用于模拟退火时的概率计算
rannum2 = 1;                %   随机数2，用于对CNC机器的故障判定
rannum3 = 1;                %   随机数3，用于确定故障发生的时间
Accrecord = zeros(10, 3);   %   故障记录矩阵
Acccount = 0;               %   故障记录

while t < Twork
    %   根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
        case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
```

```
            Trgvm(8) = tm2;
        case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
      case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
         Trgvm(7) = 0;
            Trgvm(8) = 0;
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
   Trgvw(3) = trwo;
    Trgvw(5) = trwo;
   Trgvw(7) = trwo;
   Trgvw(2) = trwe;
   Trgvw(4) = trwe;
   Trgvw(6) = trwe;
   Trgvw(8) = trwe;
    %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
    if pawsecond == 0
        Trgvw(2) = Trgvw(2) + 100000;
        Trgvw(4) = Trgvw(4) + 100000;
        Trgvw(6) = Trgvw(6) + 100000;
        Trgvw(8) = Trgvw(8) + 100000;
    else
        Trgvw(1) = Trgvw(1) + 100000;
        Trgvw(3) = Trgvw(3) + 100000;
        Trgvw(5) = Trgvw(5) + 100000;
        Trgvw(7) = Trgvw(7) + 100000;
    end
    %   计算总时间矩阵
    Ttotal = Trgvm + Trgvw + Tcncw;
    %   找出最短路径
    %   基于模拟退火算法生成最短路径位置
    rannum1 = rand(1);
    if rannum1 > 0.02
```

```
        [tmin, minPos] = min(Ttotal);
    else
        [sortTtotal, sortix] = sort(Ttotal);
        tmin = Ttotal(sortix(2));
        minPos = sortix(2);
    end
    %   若机械爪上有待清洗的熟料，计算清洗时间
    if pawclear > 0
        Tclear = tclr;
    else
        Tclear = 100000;
    end
    %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
    if tmin > Tclear
        t = t + tclr;
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - tclr;
        pawclear = pawclear - 1;
    %   若下一步要操作的对象不在此处，则移动RGV
    elseif ceil(minPos/2) ~= Pos
        Pos = ceil(minPos/2);
        t = t + Trgvm(minPos);
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvm(minPos);
    %   若下一步要操作的对向在此处
    else
        switch minPos
            %   若操作对象为第一道工序CNC
            case {1, 3, 5, 7}
                %   若操作对象未开始工作，进行上料处理并计数
                if CNCw(minPos) == 0
                    count1(minPos) = count1(minPos) + 1;
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    %   根据概率计算机器是否发生故障，
                    %   并生成故障发生时间以及人工修复时间
                    rannum2 = rand(1);
                    if rannum2 > 0.01
                        Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                        CNCw(minPos) = 1;
                    else
                        rannum3 = ceil(rand(1) * tcnc1);
                        Acccount = Acccount + 1;
                        Accrecord(Acccount, 1) = minPos;
                        Accrecord(Acccount, 2) = t + rannum3;
                        Accrecord(Acccount, 3) = t + rannum3 + ...
                            ceil((10+1000*rannum2)*60);
```

```
                    CNCw(minPos) = 2;
                    Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
                end
            %   若操作对象的工作状态标记大于1
            else
                %   若操作对象已工作完毕，则先下料再上料并计数
                if Tcncw(minPos) == 0
                    if CNCw(minPos) == 1
                        endtime1(count1(minPos), minPos) = t;
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawsecond = pawsecond + 1;
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc1);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                    %   若状态标记为2，说明上一次运行时故障，所以直接上料
                    else
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc1);
```

```
                        Acccount = Acccount + 1;
                        Accrecord(Acccount, 1) = minPos;
                        Accrecord(Acccount, 2) = t + rannum3;
                        Accrecord(Acccount, 3) = t + rannum3 + ...
                            ceil((10+1000*rannum2)*60);
                        CNCw(minPos) = 2;
                        Tcncw(minPos) = rannum3 + ...
                            ceil((10+1000*rannum2)*60);
                    end
                end
            %   若操作对象未工作完毕，则等待
            else
                t = t + 1;
                Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
            end
        end
    %   若操作对象为第二道工序CNC
    case {2, 4, 6, 8}
        %   若操作对象未开始工作，上料并计数
        if CNCw(minPos) == 0
            count2(minPos) = count2(minPos) + 1;
            starttime2(count2(minPos), minPos) = t;
            t = t + Trgvw(minPos);
            Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
            Tcncw(Tcncw<0) = 0;
            %   根据概率计算机器是否发生故障，
            %   并生成故障发生时间以及人工修复时间
            rannum2 = rand(1);
            if rannum2 > 0.01
                Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                CNCw(minPos) = 1;
            else
                rannum3 = ceil(rand(1) * tcnc2);
                Acccount = Acccount + 1;
                Accrecord(Acccount, 1) = minPos;
                Accrecord(Acccount, 2) = t + rannum3;
                Accrecord(Acccount, 3) = t + rannum3 + ...
                    ceil((10+1000*rannum2)*60);
                CNCw(minPos) = 2;
                Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
            end
            pawsecond = pawsecond - 1;
        %   若操作对象工作状态标记大于1
        else
            %   若操作对象已工作完毕
            if Tcncw(minPos) == 0
                %   且状态标记为1，则先下料再上料并计数
```

```
                    if CNCw(minPos) == 1
                        endtime2(count2(minPos), minPos) = t;
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawclear = pawclear + 1;
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc2);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                        pawsecond = pawsecond - 1;
                    %   若状态标记为2，则说明上一次工作时故障，直接上料
                    else
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc2);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
```

```
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                        pawsecond = pawsecond - 1;
                    end
                %   若操作对象未工作完毕，则等待
                else
                    t = t + 1;
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
                end
            end
        end
    end
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   除此之外，由于该模型在寻找最短路径时采用了模拟退火算法，因此得到的结果
%   与Excel可能有一定出入

%   由于机器的损坏也具有一定概率，所以得到的结果与Excel可能有一定出入

%   判断每个产出物料是否有效
logiccount = endtime2 > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
%   分第一道工序和第二道工序
starttime1
starttime2
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   分第一道工序和第二道工序，endtime为0说明该物料加工出现故障
endtime1
endtime2
%   输出故障记录
Accrecord
%   输出总产量
TotalProduct = sum(logiccount(:))
```

附录 H 两道工序 (第 2 组数据，有故障)–MATLAB 源程序

Extension = .otf ,

UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%  MATLAB初始化
clear
clc

%  数据初始化
tm1 = 23;                   %  RGV移动1个单位时间
tm2 = 41;                   %  RGV移动2个单位时间
tm3 = 59;                   %  RGV移动3个单位时间
tcnc1 = 280;                %  CNC完成第一道工序所需时间
tcnc2 = 500;                %  CNC完成第二道工序所需时间
trwo = 30;                  %  RGV为奇数CNC上下料时间
trwe = 35;                  %  RGV为偶数CNC上下料时间
tclr = 30;                  %  RGV清洗熟料时间
Twork = 28800;              %  总工作时间
CNCnum = 8;                 %  CNC机器数
t = 0;                      %  时间初始化
Pos = 1;                    %  位置初始化
CNCw = zeros(1, CNCnum);    %  CNC工作状态标志
Trgvm = zeros(1, CNCnum);     %  RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);   %  RGV工作时间矩阵
Tcncw = zeros(1, CNCnum);   %  CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);  %  总时间矩阵
pawsecond = 0;              %  需要进行第二道工序的物料
pawclear = 0;               %  需要清洗的物料
Tclear = 100000;            %  清洗剩余时间
count1 = zeros(1, CNCnum);  %  计算每台机器第一道工序上料数目
count2 = zeros(1, CNCnum);  %  计算每台机器第二道工序上料数目
starttime1 = zeros(60, CNCnum);% 每台机器第一道工序上料所对应时间
starttime2 = zeros(60, CNCnum);% 每台机器第二道工序上料所对应时间
endtime1 = zeros(60, CNCnum); % 每台机器第一道工序下料所对应时间
endtime2 = zeros(60, CNCnum); % 每台机器第二道工序下料所对应时间
sortTtotal = zeros(1, CNCnum); % 模拟退火算法所需的排序矩阵
sortix = zeros(1, CNCnum);  %  模拟退火算法所需的位置矩阵
rannum1 = 1;                %  随机数1，用于模拟退火时的概率计算
rannum2 = 1;                %  随机数2，用于对CNC机器的故障判定
rannum3 = 1;                %  随机数3，用于确定故障发生的时间
Accrecord = zeros(10, 3);   %  故障记录矩阵
Acccount = 0;               %  故障记录

while t < Twork
    %  根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
```

```
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
        case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
       case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
     case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
         Trgvm(7) = 0;
            Trgvm(8) = 0;
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
   Trgvw(3) = trwo;
    Trgvw(5) = trwo;
   Trgvw(7) = trwo;
   Trgvw(2) = trwe;
   Trgvw(4) = trwe;
   Trgvw(6) = trwe;
   Trgvw(8) = trwe;
    %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
    if pawsecond == 0
        Trgvw(1) = Trgvw(1) + 100000;
        Trgvw(3) = Trgvw(3) + 100000;
```

```
        Trgvw(5) = Trgvw(5) + 100000;
        Trgvw(7) = Trgvw(7) + 100000;
    else
        Trgvw(2) = Trgvw(2) + 100000;
        Trgvw(4) = Trgvw(4) + 100000;
        Trgvw(6) = Trgvw(6) + 100000;
        Trgvw(8) = Trgvw(8) + 100000;
    end
    %   计算总时间矩阵
    Ttotal = Trgvm + Trgvw + Tcncw;
    %   找出最短路径
    %   基于模拟退火算法生成最短路径位置
    rannum1 = rand(1);
    if rannum1 > 0.02
        [tmin, minPos] = min(Ttotal);
    else
        [sortTtotal, sortix] = sort(Ttotal);
        tmin = Ttotal(sortix(2));
        minPos = sortix(2);
    end
    %   若机械爪上有待清洗的熟料，计算清洗时间
    if pawclear > 0
        Tclear = tclr;
    else
        Tclear = 100000;
    end
    %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
    if tmin > Tclear
        t = t + tclr;
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - tclr;
        pawclear = pawclear - 1;
    %   若下一步要操作的对象不在此处，则移动RGV
    elseif ceil(minPos/2) ~= Pos
        Pos = ceil(minPos/2);
        t = t + Trgvm(minPos);
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvm(minPos);
    %   若下一步要操作的对向在此处
    else
        switch minPos
            %   若操作对象为第一道工序CNC
            case {2, 4, 6, 8}
                %   若操作对象未开始工作，进行上料处理并计数
                if CNCw(minPos) == 0
                    count1(minPos) = count1(minPos) + 1;
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
```

```
                Tcncw(Tcncw<0) = 0;
                %   根据概率计算机器是否发生故障，
                %   并生成故障发生时间以及人工修复时间
                rannum2 = rand(1);
                if rannum2 > 0.01
                    Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                    CNCw(minPos) = 1;
                else
                    rannum3 = ceil(rand(1) * tcnc1);
                    Acccount = Acccount + 1;
                    Accrecord(Acccount, 1) = minPos;
                    Accrecord(Acccount, 2) = t + rannum3;
                    Accrecord(Acccount, 3) = t + rannum3 + ...
                        ceil((10+1000*rannum2)*60);
                    CNCw(minPos) = 2;
                    Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
                end
            %   若操作对象的工作状态标记大于1
            else
                %   若操作对象已工作完毕，则先下料再上料并计数
                if Tcncw(minPos) == 0
                    if CNCw(minPos) == 1
                        endtime1(count1(minPos), minPos) = t;
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawsecond = pawsecond + 1;
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc1);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                    %   若状态标记为2，说明上一次运行时故障，所以直接上料
```

```
                    else
                        count1(minPos) = count1(minPos) + 1;
                        starttime1(count1(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc1);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                    end
                %   若操作对象未工作完毕，则等待
                else
                    t = t + 1;
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
                end
            end
        %   若操作对象为第二道工序CNC
        case {1, 3, 5, 7}
            %   若操作对象未开始工作，上料并计数
            if CNCw(minPos) == 0
                count2(minPos) = count2(minPos) + 1;
                starttime2(count2(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                %   根据概率计算机器是否发生故障，
                %   并生成故障发生时间以及人工修复时间
                rannum2 = rand(1);
                if rannum2 > 0.01
                    Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                    CNCw(minPos) = 1;
                else
                    rannum3 = ceil(rand(1) * tcnc2);
```

```
                    Acccount = Acccount + 1;
                    Accrecord(Acccount, 1) = minPos;
                    Accrecord(Acccount, 2) = t + rannum3;
                    Accrecord(Acccount, 3) = t + rannum3 + ...
                        ceil((10+1000*rannum2)*60);
                    CNCw(minPos) = 2;
                    Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
                end
                pawsecond = pawsecond - 1;
            %   若操作对象工作状态标记大于1
            else
                %   若操作对象已工作完毕
                if Tcncw(minPos) == 0
                    %   且状态标记为1，则先下料再上料并计数
                    if CNCw(minPos) == 1
                        endtime2(count2(minPos), minPos) = t;
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawclear = pawclear + 1;
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc2);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                        pawsecond = pawsecond - 1;
                    %   若状态标记为2，则说明上一次工作时故障，直接上料
                    else
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
```

```
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc2);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                        pawsecond = pawsecond - 1;
                    end
                %   若操作对象未工作完毕，则等待
                else
                    t = t + 1;
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
                end
            end
        end
    end
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   除此之外，由于该模型在寻找最短路径时采用了模拟退火算法，因此得到的结果
%   与Excel可能有一定出入

%   由于机器的损坏也具有一定概率，所以得到的结果与Excel可能有一定出入

%   判断每个产出物料是否有效
logiccount = endtime2 > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
%   分第一道工序和第二道工序
starttime1
starttime2
```

```
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   分第一道工序和第二道工序，endtime为0说明该物料加工出现故障
endtime1
endtime2
%   输出故障记录
Accrecord
%   输出总产量
TotalProduct = sum(logiccount(:))
```

附录 I 两道工序 (第 3 组数据，有故障)–MATLAB 源程序

Extension = .otf ,
UprightFont = Inconsolatazi4-Regular,
BoldFont = Inconsolatazi4-Bold

```
%   MATLAB初始化
clear
clc

%   数据初始化
tm1 = 18;                   %   RGV移动1个单位时间
tm2 = 32;                   %   RGV移动2个单位时间
tm3 = 46;                   %   RGV移动3个单位时间
tcnc1 = 455;                %   CNC完成第一道工序所需时间
tcnc2 = 182;                %   CNC完成第二道工序所需时间
trwo = 27;                  %   RGV为奇数CNC上下料时间
trwe = 32;                  %   RGV为偶数CNC上下料时间
tclr = 25;                  %   RGV清洗熟料时间
Twork = 28800;              %   总工作时间
CNCnum = 8;                 %   CNC机器数
t = 0;                      %   时间初始化
Pos = 1;                    %   位置初始化
CNCw = zeros(1, CNCnum);    %   CNC工作状态标志
Trgvm = zeros(1, CNCnum);    %   RGV移动时间矩阵
Trgvw = zeros(1, CNCnum);   %   RGV工作时间矩阵
Tcncw = zeros(1, CNCnum);   %   CNC工作时间矩阵
Ttotal = zeros(1, CNCnum);  %   总时间矩阵
pawsecond = 0;              %   需要进行第二道工序的物料
pawclear = 0;               %   需要清洗的物料
Tclear = 100000;            %   清洗剩余时间
count1 = zeros(1, CNCnum);  %   计算每台机器第一道工序上料数目
count2 = zeros(1, CNCnum);  %   计算每台机器第二道工序上料数目
starttime1 = zeros(70, CNCnum);% 每台机器第一道工序上料所对应时间
starttime2 = zeros(70, CNCnum);% 每台机器第二道工序上料所对应时间
endtime1 = zeros(70, CNCnum); % 每台机器第一道工序下料所对应时间
endtime2 = zeros(70, CNCnum); % 每台机器第二道工序下料所对应时间
```

```
sortTtotal = zeros(1, CNCnum); % 模拟退火算法所需的排序矩阵
sortix = zeros(1, CNCnum);   %  模拟退火算法所需的位置矩阵
rannum1 = 1;                 %  随机数1，用于模拟退火时的概率计算
rannum2 = 1;                 %  随机数2，用于对CNC机器的故障判定
rannum3 = 1;                 %  随机数3，用于确定故障发生的时间
Accrecord = zeros(10, 3);    %  故障记录矩阵
Acccount = 0;                %  故障记录

while t < Twork
    %  根据RGV所在位置计算RGV移动时间矩阵
    switch Pos
        case 1
            Trgvm(1) = 0;
            Trgvm(2) = 0;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = tm2;
            Trgvm(6) = tm2;
            Trgvm(7) = tm3;
            Trgvm(8) = tm3;
         case 2
            Trgvm(1) = tm1;
            Trgvm(2) = tm1;
            Trgvm(3) = 0;
            Trgvm(4) = 0;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
            Trgvm(7) = tm2;
            Trgvm(8) = tm2;
        case 3
            Trgvm(1) = tm2;
            Trgvm(2) = tm2;
            Trgvm(3) = tm1;
            Trgvm(4) = tm1;
            Trgvm(5) = 0;
            Trgvm(6) = 0;
            Trgvm(7) = tm1;
            Trgvm(8) = tm1;
       case 4
            Trgvm(1) = tm3;
            Trgvm(2) = tm3;
            Trgvm(3) = tm2;
            Trgvm(4) = tm2;
            Trgvm(5) = tm1;
            Trgvm(6) = tm1;
         Trgvm(7) = 0;
            Trgvm(8) = 0;
```

```
    end
    %   根据RGV要对设备进行的操作计算RGV工作时间矩阵
    Trgvw(1) = trwo;
   Trgvw(3) = trwo;
    Trgvw(5) = trwo;
   Trgvw(7) = trwo;
   Trgvw(2) = trwe;
   Trgvw(4) = trwe;
   Trgvw(6) = trwe;
   Trgvw(8) = trwe;
    %   若暂时没有第一道工序加工完成的半成品，则第二道工序对应机器暂停
    if pawsecond == 0
        Trgvw(2) = Trgvw(2) + 100000;
        Trgvw(5) = Trgvw(5) + 100000;
        Trgvw(7) = Trgvw(7) + 100000;
    else
        Trgvw(1) = Trgvw(1) + 100000;
        Trgvw(3) = Trgvw(3) + 100000;
        Trgvw(4) = Trgvw(4) + 100000;
        Trgvw(6) = Trgvw(6) + 100000;
        Trgvw(8) = Trgvw(8) + 100000;
    end
    %   计算总时间矩阵
    Ttotal = Trgvm + Trgvw + Tcncw;
    %   找出最短路径
    %   基于模拟退火算法生成最短路径位置
    rannum1 = rand(1);
    if rannum1 > 0.02
        [tmin, minPos] = min(Ttotal);
    else
        [sortTtotal, sortix] = sort(Ttotal);
        tmin = Ttotal(sortix(2));
        minPos = sortix(2);
    end
    %   若机械爪上有待清洗的熟料，计算清洗时间
    if pawclear > 0
        Tclear = tclr;
    else
        Tclear = 100000;
    end
    %   若下一步最短路径所对应时间大于清洗时间，先进行清洗操作
    if tmin > Tclear
        t = t + tclr;
        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - tclr;
        pawclear = pawclear - 1;
    %   若下一步要操作的对象不在此处，则移动RGV
    elseif ceil(minPos/2) ~= Pos
```

```
126         Pos = ceil(minPos/2);
127         t = t + Trgvm(minPos);
128         Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvm(minPos);
129     %   若下一步要操作的对向在此处
130     else
131         switch minPos
132             %   若操作对象为第一道工序CNC
133             case {1, 3, 4, 6, 8}
134                 %   若操作对象未开始工作，进行上料处理并计数
135                 if CNCw(minPos) == 0
136                     count1(minPos) = count1(minPos) + 1;
137                     starttime1(count1(minPos), minPos) = t;
138                     t = t + Trgvw(minPos);
139                     Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
140                     Tcncw(Tcncw<0) = 0;
141                     %   根据概率计算机器是否发生故障，
142                     %   并生成故障发生时间以及人工修复时间
143                     rannum2 = rand(1);
144                     if rannum2 > 0.01
145                         Tcncw(minPos) = Tcncw(minPos) + tcnc1;
146                         CNCw(minPos) = 1;
147                     else
148                         rannum3 = ceil(rand(1) * tcnc1);
149                         Acccount = Acccount + 1;
150                         Accrecord(Acccount, 1) = minPos;
151                         Accrecord(Acccount, 2) = t + rannum3;
152                         Accrecord(Acccount, 3) = t + rannum3 + ...
153                             ceil((10+1000*rannum2)*60);
154                         CNCw(minPos) = 2;
155                         Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
156                     end
157                 %   若操作对象的工作状态标记大于1
158                 else
159                     %   若操作对象已工作完毕，则先下料再上料并计数
160                     if Tcncw(minPos) == 0
161                         if CNCw(minPos) == 1
162                             endtime1(count1(minPos), minPos) = t;
163                             count1(minPos) = count1(minPos) + 1;
164                             starttime1(count1(minPos), minPos) = t;
165                             t = t + Trgvw(minPos);
166                             pawsecond = pawsecond + 1;
167                             Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
168                             Tcncw(Tcncw<0) = 0;
169                             %   根据概率计算机器是否发生故障，
170                             %   并生成故障发生时间以及人工修复时间
171                             rannum2 = rand(1);
172                             if rannum2 > 0.01
```

```
                        Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                        CNCw(minPos) = 1;
                    else
                        rannum3 = ceil(rand(1) * tcnc1);
                        Acccount = Acccount + 1;
                        Accrecord(Acccount, 1) = minPos;
                        Accrecord(Acccount, 2) = t + rannum3;
                        Accrecord(Acccount, 3) = t + rannum3 + ...
                            ceil((10+1000*rannum2)*60);
                        CNCw(minPos) = 2;
                        Tcncw(minPos) = rannum3 + ...
                            ceil((10+1000*rannum2)*60);
                    end
                %   若状态标记为2，说明上一次运行时故障，所以直接上料
                else
                    count1(minPos) = count1(minPos) + 1;
                    starttime1(count1(minPos), minPos) = t;
                    t = t + Trgvw(minPos);
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                    Tcncw(Tcncw<0) = 0;
                    %   根据概率计算机器是否发生故障，
                    %   并生成故障发生时间以及人工修复时间
                    rannum2 = rand(1);
                    if rannum2 > 0.01
                        Tcncw(minPos) = Tcncw(minPos) + tcnc1;
                        CNCw(minPos) = 1;
                    else
                        rannum3 = ceil(rand(1) * tcnc1);
                        Acccount = Acccount + 1;
                        Accrecord(Acccount, 1) = minPos;
                        Accrecord(Acccount, 2) = t + rannum3;
                        Accrecord(Acccount, 3) = t + rannum3 + ...
                            ceil((10+1000*rannum2)*60);
                        CNCw(minPos) = 2;
                        Tcncw(minPos) = rannum3 + ...
                            ceil((10+1000*rannum2)*60);
                    end
                end
            %   若操作对象未工作完毕，则等待
            else
                t = t + 1;
                Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
            end
        end
    %   若操作对象为第二道工序CNC
    case {2, 5, 7}
        %   若操作对象未开始工作，上料并计数
```

```
            if CNCw(minPos) == 0
                count2(minPos) = count2(minPos) + 1;
                starttime2(count2(minPos), minPos) = t;
                t = t + Trgvw(minPos);
                Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                Tcncw(Tcncw<0) = 0;
                %   根据概率计算机器是否发生故障，
                %   并生成故障发生时间以及人工修复时间
                rannum2 = rand(1);
                if rannum2 > 0.01
                    Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                    CNCw(minPos) = 1;
                else
                    rannum3 = ceil(rand(1) * tcnc2);
                    Acccount = Acccount + 1;
                    Accrecord(Acccount, 1) = minPos;
                    Accrecord(Acccount, 2) = t + rannum3;
                    Accrecord(Acccount, 3) = t + rannum3 + ...
                        ceil((10+1000*rannum2)*60);
                    CNCw(minPos) = 2;
                    Tcncw(minPos) = rannum3 + ceil((10+1000*rannum2)*60);
                end
                pawsecond = pawsecond - 1;
            %   若操作对象工作状态标记大于1
            else
                %   若操作对象已工作完毕
                if Tcncw(minPos) == 0
                    %   且状态标记为1，则先下料再上料并计数
                    if CNCw(minPos) == 1
                        endtime2(count2(minPos), minPos) = t;
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        pawclear = pawclear + 1;
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc2);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
```

```
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                        pawsecond = pawsecond - 1;
                    %   若状态标记为2，则说明上一次工作时故障，直接上料
                    else
                        count2(minPos) = count2(minPos) + 1;
                        starttime2(count2(minPos), minPos) = t;
                        t = t + Trgvw(minPos);
                        Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - Trgvw(minPos);
                        Tcncw(Tcncw<0) = 0;
                        %   根据概率计算机器是否发生故障，
                        %   并生成故障发生时间以及人工修复时间
                        rannum2 = rand(1);
                        if rannum2 > 0.01
                            Tcncw(minPos) = Tcncw(minPos) + tcnc2;
                            CNCw(minPos) = 1;
                        else
                            rannum3 = ceil(rand(1) * tcnc2);
                            Acccount = Acccount + 1;
                            Accrecord(Acccount, 1) = minPos;
                            Accrecord(Acccount, 2) = t + rannum3;
                            Accrecord(Acccount, 3) = t + rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                            CNCw(minPos) = 2;
                            Tcncw(minPos) = rannum3 + ...
                                ceil((10+1000*rannum2)*60);
                        end
                        pawsecond = pawsecond - 1;
                    end
                %   若操作对象未工作完毕，则等待
                else
                    t = t + 1;
                    Tcncw(CNCw >= 1) = Tcncw(CNCw >= 1) - 1;
                end
            end
        end
    end
    Tcncw(Tcncw<0) = 0;
end

%   数据输出

%   由于最后快接近八小时的时候需要停止某些机器和RGV的上下料操作以保证能在八小时
```

```
%   内进行设备停止并回到初始位置，因此程序计算结果需经进一步人工处理才能得到
%   真正的值，所以程序生成的数据和导入Excel表格中的数据会有微小误差

%   除此之外，由于该模型在寻找最短路径时采用了模拟退火算法，因此得到的结果
%   与Excel可能有一定出入

%   由于机器的损坏也具有一定概率，所以得到的结果与Excel可能有一定出入

%   判断每个产出物料是否有效
logiccount = endtime2 > 0;
%   输出每个物料上料时间，其中矩阵的行为不同的物料，列为在哪个CNC机器上
%   分第一道工序和第二道工序
starttime1
starttime2
%   输出每个物料下料时间，其中矩阵的行为不同的物料。列为在哪个机器上
%   分第一道工序和第二道工序，endtime为0说明该物料加工出现故障
endtime1
endtime2
%   输出故障记录
Accrecord
%   输出总产量
TotalProduct = sum(logiccount(:))
```