基于多目标规划的智能 RGV 的动态调度策略

摘要

本文根据题目给定的智能加工系统及系统作业参数，针对一道工序、两道工序、一道工序故障处理、两道工序故障处理四种作业场景，以系统作业总耗时最小化、成料总产出最大化作为优化目标，建立数学模型，并给出相应的 RGV 最佳调度策略，最后对模型效果做出检验。

针对一道工序物料加工作业情况，建立了**多目标规划**模型，解决了一道工序情况下的 RGV 动态调度问题。首先，利用**枚举法**列举由 4 对 CNC 所构成的固定运行回路，分析各回路耗时，确定出最优路径: **1→2→3→4→7→8→5→6**。之后，确定 RGV 的等待时间、旅行时间、上下料时间、加工时间及清洗作业时间。然后，以系统总耗时、成料总产出作为决策变量，建立**多目标规划**模型。继而使用**迭代法**求解出该策略下各物料的上下料开始时间及三组参数情景下的最终产出量，三组参数下的产出量分别为：**383**、**360**、**392**。

针对两道工序物料加工作业情况，建立了 **0-1 规划**模型，解决了两道工序情况下的 RGV 动态调度问题。首先，考虑到 CNC 加工完成两道工序所需时间的不同以及 CNC 位置分布，按两道工序加工时间比例并适当调整，以 **4:4**、**3:5**、**5:3** 为 CNC 分配刀具。然后，引入一个 **0-1 变量**以确定当前物料所在工序，区分两种加工用时，进而在模型一基础上，建立 **0-1 规划模型**。之后利用**动态规划**算法对模型进行求解，得出最终调度策略。最后，将三组数据代入模型计算，得出三组产出量分别为 **255**、**227**、**232**。

针对一道工序故障加工作业情况，建立基于**空闲时间插入法**的故障扰动模型，解决了一道工序有故障存在情况下的 RGV 动态调度问题。首先，根据故障发生概率和故障间隔时间分布，基于情况一的模型于系统总耗时公式中插入一项由排除故障所增加的 RGV 额外等待时间，建立基于**空闲时间插入法**的故障扰动模型。之后，在情况一的求解算法中增加对 CNC 故障的判断步骤和故障处理时间插入更新机制，求解 RGV 动态调度策略。最后，代入三组数据得出最大产出量分别为 **373**、**343**、**378**。

针对两道工序故障加工作业情况，建立两道工序下基于**空闲时间插入法**的 **0-1 规划**模型。首先，以情况二的模型为基础，在系统总耗时公式中插入由排除故障所增加的 RGV 额外等待时间，建立基于空闲时间插入法的 **0-1 规划**模型。其次，在算法二基础上增加故障处理更新机制，得出该情况下的求解算法。最后，将三组数据代入模型计算得出最终成料产出量分别为 **237**、**218**、**231**，三组数据下加工系统的产率分别为 **0.008371**、**0.007641**、**0.007821**，验证了模型的正确性。

关键词：多目标规划　迭代法　动态规划　0-1 规划　空闲时间插入法

一、问题重述

1.1. 问题背景

随着计算机集成技术、机械自动化、自动化立体仓库等技术的飞跃式发展，制造加工行业对智能性、灵活性的要求越来越高，制定符合车间生产情况的、最优化的智能制造动态调度能大幅度提高车间生产效率、降低产品生产成本，实现生产车间的整体利益最大化。RGV 智能系统的动态调度是智能系统应用的重中之重。

本题的智能加工系统包括 8 台 CNC、1 台带机械手臂和手爪的 RGV、1 条 RGV 直线轨道、1 条上料传送带和 1 条下料传送带等附属设备，每台 CNC 同一时间只能安装 1 种刀具加工 1 个物料完成 1 道工序。

系统启动后，RGV 在 CNC1#和 CNC2#正中间的初始位置，所有 CNC 都处于空闲状态，均向 RGV 发出上料需求信号。否则处于加工作业状态，在加工作业完成即刻向 RGV 发出需求信号。在接到信号后，RGV 移动至指定 CNC 处，交替利用两只手爪进行上下料，继而取出清洗槽中的成料、转动手爪、放入熟料进行清洗作业，并将成料放至下料传送带送出系统。RGV 在完成一项作业任务后，若未接到其他作业指令，则在原地等待执行下一指令。CNC 完成一个物料的加工作业任务后，若有 RGV 即刻到达为其上下料，则该 CNC 即刻向 RGV 发出需求信号，否则等待。系统循环进行上下料和清洗作业，直到停止，RGV 返回初始位置。

题中智能加工系统的作业场景分为以下四种情况：

1）情况一：一道工序的物料加工作业；

2）情况二：两道工序的物料加工作业；

3）情况三：一道工序的物料加工作业发生故障；

4）情况四：两道工序的物料加工作业发生故障。

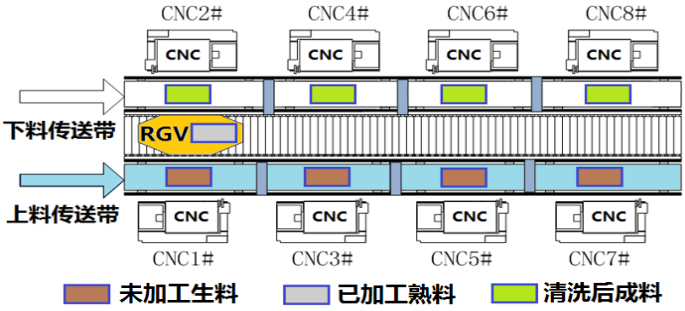

图 1-1 智能加工系统示意图

1.2. 问题要求

任务 1：对一般问题进行研究，给出 RGV 动态调度模型和相应的求解算法；

任务 2：利用表 1 中系统作业参数的 3 组数据分别检验模型的实用性和算法的有效性，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。

二、问题分析

2.1. 一道工序的分析

对于情况一，首先，考虑到 CNC 安装刀具的统一性，将 8 台 CNC 视为完全相同的 8 个固定服务点，每两台 CNC 之间可以通过 RGV 连接，实现对系统构成的简化。然后利用枚举法列举出由同列排放的 4 个 CNC 组合所构成的运行回路，计算各回路耗时，确定出最优路径。之后建立多目标规划模型，使用迭代法求解出三组参数场景下的最高产出量。

2.2. 两道工序的分析

对于情况二，通过分析表中数据可得，RGV 的运行时间相对 CNC 加工时间很短,所以不同类型 CNC 的数量对系统效率的影响要远大于 CNC 分布的影响。因此主要以不同工序的加工时间比例来确定不同 CNC 的刀具分配比例，同时适当考虑 CNC 位置分布并对分配比例进行微调。

考虑到 CNC 加工完成两道工序不同情况，因此引入一个 0-1 变量，用以确定当前物料所处工序。又考虑到 RGV 的路径动态地取决于 RGV 和 CNC 的状态和位置，所以用 RGV 动态路径的旅行时间代表其路径长度，利用动态规划的思想进行路径选择，建立基于 0-1 规划的 RGV 调度模型。并且利用 MATLAB 编写对应程序，最后求解出 RGV 最佳调度策略。

2.3. 一道工序故障情况的分析

对于情况三，首先，考虑到 CNC 故障排除时间对系统总耗时造成的不良影响，以及故障发生与否的不确定性、故障排除用时的波动性，故生成一个随机空闲时间，插入情况一模型的系统耗时公式中，建立一个基于空闲时间插入法的故障扰动模型。之后，在情况一的求解算法中增加对 CNC 故障的判断步骤和故障处理时间插入更新机制，求解 RGV 最佳调度策略。

2.4. 两道工序故障情况的分析

对于情况四，其将情况二与情况三结合在一起，因此在模型二与模型三的基础上，建立基于空闲时间插入法的 0-1 规划模型。针对模型的求解，在算法二的基础上增加故障处理更新机制，判断 CNC 的工作状态，得出该情况下的求解算法，求解 RGV 最佳调度策略。

三、模型假设

1.假设上下料传送带的连动或独立运动对模型无影响；

2. RGV 运行至需要作业的某 CNC 处时，上料传送带将生料同时送到该 CNC 正前方，下料传送带将清洗后的成料立刻送走；

3.假设智能加工系统中的所有传感器反应时间很短，可忽略不计；

4.假设故障时间间隔分布服从泊松分布。

四、符号说明

符号	意义
i	迭代次数
$T_{\min}$	一轮加工的最优总用时（s）
M	一个班次的成料总产出
t_{pk}	CNC 加工第k道工序的用时
C_{ij}	第i次迭代第j个物料所在工序
T_i	第i轮迭代的系统总耗时
T_l	RGV 旅行时间
T_z	RGV 上下料作业时间
T_j	CNC 加工时间
T_q	RGV 清洗作业时间
T_d	CNC 等待时间

五、模型建立与求解

5.1. 情况一模型的建立与求解

5.1.1. RGV 运行路径的确定

在无故障的情况下，如果 RGV 每次移动后只为一台 CNC 上下料，则其运行回路的耗时会更多，为尽可能地减小 RGV 的旅行时间,我们筛选出 RGV 不需要移动只需旋转手爪便能更换机床的 4 组同列排放的 CNC 设备，即(CNC1#,CNC2#)(CNC3#,CNC4#)(CNC5#,CNC6#)(CNC7#,CNC8#)，由于 RGV 的

初始位置在CNC#1处，故RGV首先利用的CNC设备对必然是(CNC1#,CNC2#)。

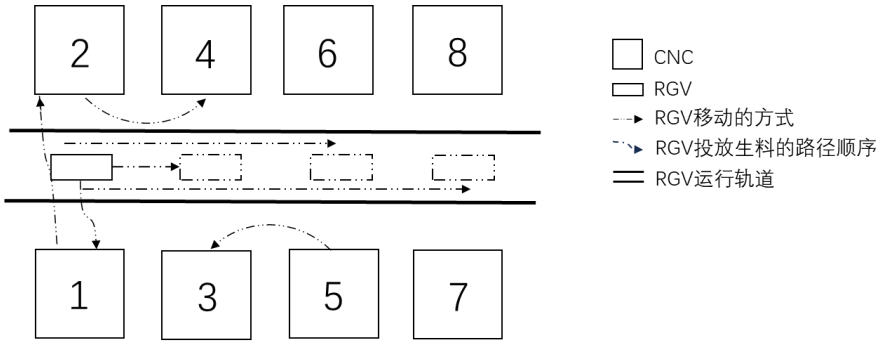

图 5-1 RGV 工作路径的类型及物料需求点 CNC

如果分别用t_1、t_2、t_3表示 RGV 移动 1 个单位、2 个单位和 3 个单位所需要的时间，且有$t_1 < t_2 < t_3, 2t_1 \geq t_2, 3t_1 \geq t_3$，则我们可利用枚举法，列举出有限的 6 种 RGV 可能的周期性运行回路：

回路 1:(1，2)→(3，4)→(7，8)→(5，6)→(1，2)，**共耗时**$2t_1 + 2t_2$；

回路 2:(1，2)→(5，6)→(7，8)→(3，4)→(1，2)，**共耗时**$2t_1 + 2t_2$；

回路 3:(1，2)→(3，4)→(5，6)→(7，8)→(1，2)，**共耗时**$3t_1 + t_3$；

回路 4:(1，2)→(7，8)→(5，6)→(3，4)→(1，2)，**共耗时**$3t_1 + t_3$；

回路 5:(1，2)→(7，8)→(3，4)→(5，6)→(1，2)，**共耗时**$t_1 + 2t_2 + t_3$；

回路 6:(1，2)→(5，6)→(3，4)→(7，8)→(1，2)，**共耗时**$t_1 + 2t_2 + t_3$；

易知$2t_1 + 2t_2 < 3t_1 + t_3 < t_1 + 2t_2 + t_3$.所以回路 1 或回路 2 耗时最短，在这里仅以回路 1 为例。

5.1.2. 多目标规划模型的建立

多目标规划是一种用于解决多个目标函数在给定区域上的最优化问题的数学规划方法，其为同时实现多个目标，为每个目标分配一个衡量偏离目标程度的系数因子，通过平衡各目标的实现程度，使得总偏差最小。

情况一中物料加工作业仅一道工序，每台 CNC 安装同样的刀具，物料可以在任一台 CNC 上加工完成。我们需要制定策略，使得系统在一个班次内，实现最大成料产出，且在一个运行回路中的耗时尽可能小。

故使用多目标规划方法，对一道工序无故障情况下智能加工系统的最优运作流程进行建模：

1）目标函数的确定

对于加工系统的一次 8 小时运行而言，需要给出调度方案使得 CNC 总等待时间最少、成料产出量最大，

对于整个加工过程，RGV 的工作流程如下：

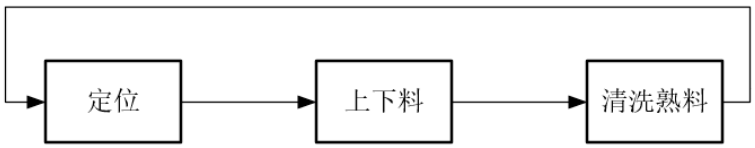

图 5-2　RGV 一次完整作业示意图

考虑 8 次如图所示的 RGV 一次完整作业，当 RGV 对某台 CNC 进行上下料时，其它 CNC 的加工可能已经完成并进行需求提示，则在 RGV 到达前，这些 CNC 都将产生等待时间；此外，根据之前的结论，RGV 会优先对同一位置的两台 CNC 依次作业。

在假设条件的基础下，RGV 动态调度需要在 RGV 对某些时间状态做出判断后开始，判断出到某台 CNC 进行工作的过程总时间最小，则到相应 CNC 进行作业。RGV 判断过程涉及的时间变量如表 5-1 所示。

表 5-1 RGV 判断时间段

时间段	判断规则
RGV 等待时间T_d	判断 8 台 CNC 加工完成的剩余时间
CNC 加工时间T_j	设置保持稳定
RGV 清洗作业时间T_q	设置保持稳定
RGV 旅行时间T_l	判断 RGV 所在位置移动到第 1-8 台 CNC 的耗时
RGV 上下料作业时间T_z	CNC 编号同奇或同偶，则时间相等；若 CNC 编号一奇一偶，则$t_{奇} > t_{偶}$

对任意一种调度方案，RGV 每进行一次作业操作，都会使当前 CNC 产生等待时间，故设这种调度方案产生的 CNC 等待总时间为T_d。

综上，建立最优化目标函数如下：

$$\begin{cases} \max\ n \\ \min\ T_i = (T_d + T_l + T_z) + T_j + T_q \end{cases} \quad (1)$$

2）初始方案的规划

由 5.1.1.可知，RGV 初始运行路径我们选择为(1,2)→(3,4)→(7,8)→(5,6)。

图 5-3 初始上料情况图

3）约束条件的确定

在一道工序无故障条件下，第i次迭代所用时间T_i,故一个班次所耗用的总时间为ΣT_i，需要不超过 8 小时，即 8*60*60 秒。

综上所述，建立一道工序无故障加工模型：

$$\max\ n$$

$$\min\ T_i = (T_d + T_l + T_z)_{\min} + T_j + T_q$$

$$\begin{cases} T_1 = 0 \\ \Sigma T_i \leqslant 8 \times 60 \times 60 \\ i = 1, 2, ..., n \end{cases} \quad (2)$$

其中，T_i代表第i轮迭代的系统总耗时，T_d代表 CNC 等待时间，T_l代表 RGV 旅行时间，T_z代表 RGV 上下料作业时间，T_j代表 CNC 加工时间，T_q代表 RGV 清洗作业时间。

（1）初始状态：RGV 位于 CNC#1 前，不论先为哪台 CNC 投放物料，初始时间总为 0；

（2）迭代方程：每一次迭代过后，RGV 得到最短时间$\min\ T_i$，迭代 n 次；

（3）最优值方程：系统一个班次的工作时间为 8h，换算时间单位为 s 得到最优值方程。

5.1.3. 求解算法的设计思路

Step1：RGV 判断通过路径所需的时间以及 CNC 的工作状态.首先，RGV 根据当前位置到下一个物料投放的每个 CNC 之间的路径距离长短和每一台 CNC 是否处于加工状态，以确定通过路径的最短时间.

Step2：每一次移动需要服从的调度要求：符合调度的要求:

$$\min\ T_i = (T_d + T_l + T_z)_{\min} + T_j + T_q \qquad (3)$$

Step3：RGV 做出调度决策后，立即移动到下一个上下料需求点.

Step4：RCV 上下料操作完成前后记录耗费的总时间，更新 CNC 的状态、物料的装载、更新系统总耗时$\Sigma\min\ T_i$.

Step5：循环条件$\Sigma T_i \leqslant 8\times 60\times 60$；若总时间超过 8h，则停止循环，输出物料最大产出以及最优路径；反之，则继续循环判断，重复 Step1-4.

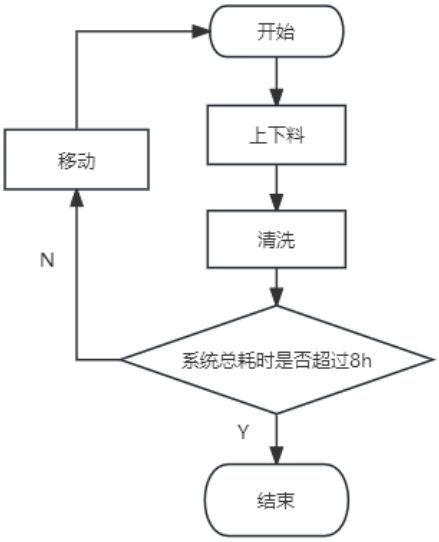

图 5-4 模型一的迭代求解流程图

5.1.4. 情况一的结论

RGV 调度策略：按照 CNC#1、CNC#2、CNC#3、CNC#4、CNC#7、CNC#8、CNC#5、CNC#6 的顺序依次为之开展作业。最终成料产量见下表。

表 5-2 一道工序最大产量

	第一组	第二组	第三组
最大产量 M（个）	383	360	392

其中，一道工序加工的第一组结果如下(详见 EXCEL 表)：

表 5-3 一道工序调度模型第一组部分数据结果表

加工物料序号	加工 CNC 编号	上料开始时间	下料开始时间
1	1	0	588
2	2	28	641
3	3	79	717
4	4	107	770

一道工序加工第二组结果如下(详见 EXCEL 表)：

表 5-4 一道工序调度模型第二组数据部分结果表

加工物料序号	加工 CNC 编号	上料开始时间	下料开始时间
1	1	0	610
2	2	30	670
3	3	88	758
4	4	118	818

一道工序加工第三组结果如下(详见 EXCEL 表)：

表 5-5 一道工序调度模型第三组数据部分结果表

加工物料序号	加工 CNC 编号	上料开始时间	下料开始时间
1	1	0	572
2	2	27	624
3	3	77	699
4	4	104	751

5.1.5. 小结

针对物料加工仅一道工序且无故障的情况，以一轮加工的最小用时和最大产出为目标函数，总加工时长不超过 8h 为约束条件，建立了多目标规划模型，解决了一道工序无故障情况下的 RGV 调度问题。

首先用枚举法筛选、列举出有限的 8 种可能运行路线，通过比较它们的耗时，得出了最佳 RGV 调度策略：按照 CNC#1→CNC#2→CNC#3→CNC#4→CNC#7→CNC#8→CNC#5→CNC#6 的顺序依次进行加工，循环这个运行回路。继而利用迭代的算法思想对模型进行求解，最终得出：截至一个班次结束前，第一组参数情境下最多能够实现 383 个成料产出，第二组参数情境下最多能够实现 360 个成料产出，第三组参数情境下最多能够实现 392 个成料产出。

图 5-5 （一道工序无故障情况）加工循环回路

5.2. 情况二模型的建立与求解

5.2.1. CNC 刀具配比的确定

由于 CNC 加工完成一个两道工序物料的两道工序所需的时间不同，故安装在 8 台 CNC 上用于加工不同工序的刀具类型就存在比例，为寻求最佳比例，分析得出，以 CNC 加工完成一个两道工序物料的两道工序分别所需的时间之比作为加工两道工序所用刀具的数量之比：

CNC 加工第一道工序用时为t_{p1}，加工第二道工序用时为t_{p2}，则确定出用于第一道工序的刀具数量（亦即 CNC 数量）为：$8\times\left[\frac{t_{p1}}{t_{p1}+t_{p2}}\right]$，用于加工第二道工序的刀具数量（亦即 CNC 数量）为$8\times\left[\frac{t_{p2}}{t_{p1}+t_{p2}}\right]$，刀具配比为$8\times\left[\frac{t_{p1}}{t_{p1}+t_{p2}}\right]:8\times\left[\frac{t_{p2}}{t_{p1}+t_{p2}}\right]$。

针对加工用时比的非整数性，对两道工序的用时均采用跨度 100 的四舍五入法，将浮点数转化为整数后再进行比例约分。

5.2.2. 0-1 规划模型的建立

0-1 规划是一种要求部分或全部决策变量是整数且只能取 0 或 1 的规划问题，常用于解决含有相互排序的约束条件的问题、固定费用问题、工件排斥等问题。

情况二中物料加工作业共两道工序，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，我们需要制定策略，使得系统在一个班次内，实现最大成料产出，且在一个运行回路中的耗时尽可能小。

由于第一道工序的熟料不需要清洗，以及两道工序的加工时间不同，故设置一个 0-1 决策变量C_{ij}表示当前物料所在工序，在此基础上对两道工序无故障情况下的智能加工系统的最有运作流程进行建模：

引入逻辑变量$C_{ij}(i,j=1,2,...,n)$，使得

$$C_{ij}=\begin{cases}0，\text{第}i\text{次迭代的第}j\text{个物料处在第一道工序}\\1，\text{第}i\text{次迭代的第}j\text{个物料处在第二道工序}\end{cases}$$

于是建立该模型的一般形式为：

$$\max\ n$$

$$\min\ T_i=(T_d+T_l+T_z)+(1-C_{ij})T_{j1}+C_{ij}T_{j2}+C_{ij}T_q$$

$$\begin{cases}T_1=0\\\Sigma T_i\leqslant 8\times 60\times 60\\C_{ij}=0\text{或}1\\i,j=1,2,...,n\end{cases}\qquad(4)$$

其中，T_i代表第i次迭代的系统总耗时，T_d代表 CNC 等待时间，T_l代表 RGV 旅行时间，T_z代表 RGV 上下料作业时间，T_j代表 CNC 加工时间，T_q代表 RGV 清洗作业时间，C_{ij}代表第i次迭代中的第j个物料所在的工序。

（1）初始状态：RGV 位于 CNC#1 前，不论先为哪台 CNC 投放物料，初始

时间总为 0；

（2）迭代方程：每一次迭代过后，RGV 得到最短时间$\min\ T_i$，迭代 n 次；

（3）最优值方程：系统一个班次的工作时间为 8h，换算时间单位为 s 得到最优值方程。

5.2.3. 动态规划算法思想

动态规划[1]问题具有以下基本特征：

1. 问题具有多阶段决策的特征。阶段可以按时间划分，也可以按空间划分。
2. 每一阶段都有相应的“状态”与之对应。
3. 每一阶段都面临一个决策，选择不同的决策将会导致下一阶段不同的状态，同时，不同的决策将会导致这一阶段不同的目标函数值。
4. 每一阶段的最优解问题可以递归地归结为下一阶段各个可能状态的最优解问题，各子问题与原问题具有完全相同的结构。能否构造这样的递推归结，是解决动态规划问题的关键。这种递推归结的过程，称为“不变嵌入”。
5. 状态具有无后效性。当某阶段状态确定后，此阶段以后过程的发展不受此阶段以前各阶段状态的影响。如下图所示：

动态规划的基本原理是将一个问题的最优解转化为求子问题的最优解，研究的对象是决策过程的最优化，其变量是流动的时间或变动的状态，最后到达整个系统最优[2]。

上文式$\min\ T_i=(T_d+T_l+T_z)+(1-C_{ij})T_{j1}+C_{ij}T_{j2}+C_{ij}T_q$即动态规划的目标函数，$\sum_{i=1}^{n}T_i\leqslant 8\times 60\times 60$则是动态规划的条件方程。

利用以上求解思想，通过递归求解最优运作路线以及成料产出量。

5.2.4. 求解算法的设计思路

Step1：按照时间比例进行刀具配给.

第一道工序 CNC 加工用时为t_{p1}，第二道工序 CNC 加工用时为t_{p2}，则配比为$8\times\left[\frac{t_{p1}}{t_{p1}+t_{p2}}\right]:8\times\left[\frac{t_{p2}}{t_{p1}+t_{p2}}\right]$.

Step2：RGV 判断通过路径所需的时间以及 CNC 的工作状态。首先，RGV 根据当前位置到下一个物料投放的每个 CNC 之间的路径距离长短和每一台 CNC 是否处于加工状态，以确定通过路径的最短时间.

Step3：每一次移动需要服从的调度要求：首先判断当前物料所在工序，若处在第一道则符合调度的要求为$\min\ T_i=(T_d+T_l+T_z)+T_{j1}$，若处在第二道则符合调度的要求为$\min\ T_i=(T_d+T_l+T_z)+T_{j2}+T_q$.

Step4：RGV 做出调度决策后，立即移动到下一个上下料需求点.

Step5：RCV 上下料操作完成前后记录耗费的总时间，更新 CNC 的状态、物料的装载、更新系统总耗时$\Sigma\min T_i$.

Step6：循环条件$\Sigma T_i\leqslant 8\times 60\times 60$；若总时间超过 8h，则停止循环，输出物料最大产出以及最优路径；反之，则继续循环判断，重复 Step2-5.

5.2.5. 情况二的结论

RGV 调度策略：

第一组参数情况下，RGV 运行顺序：CNC#2→CNC#4→CNC#8→CNC#6→$(CNC\#2\rightarrow CNC\#1\rightarrow CNC\#4\rightarrow CNC\#3\rightarrow CNC\#6\rightarrow CNC\#5\rightarrow CNC\#8\rightarrow CNC\#7)_{循环}$

第二组参数情况下，RGV 运行顺序：CNC#2→CNC#4→CNC#6→CNC#2→$(CNC\#1\longrightarrow CNC\#4\longrightarrow CNC\#3\longrightarrow CNC\#6\longrightarrow CNC\#5\longrightarrow CNC\#2)_{循环}$；

第三组参数情况下，RGV 运行顺序：CNC#1→CNC#2→CNC#3→CNC#7→CN#8→CNC#5→CN#1→CN#4→CNC#3→CNC#6→CNC#7→$(CNC\#4\rightarrow CNC\#2\rightarrow CNC\#6\rightarrow CNC\#5\rightarrow CNC\#4\rightarrow CNC\#1\rightarrow CNC\#6\rightarrow CNC\#3)_{循环}$

根据 CNC 加工完成一个两道工序的物料过程中第一和第二道工序所需的时间，得到下表的分配比例。

表 5-6 每组数据的刀具分配比例

	第一组	第二组	第三组
刀具分配比例	4:4	3:5	6:2

首先将题目给出的三组参数分别带入模型，利用 MATLAB 编程求解。经过配比检验，取其中最优的运行方案，得到第一组参数下确定使用 4:4 （4 个一号刀，4 个二号刀）的放置方案，得到 255 个成品；第二组参数下确定使用 3:5（3 个一号刀，5 个二号刀）的模型中求解，得到成品 227 个；第三组 6:2（6 个一号刀，2 个二号刀）得到成品 232 个成品。

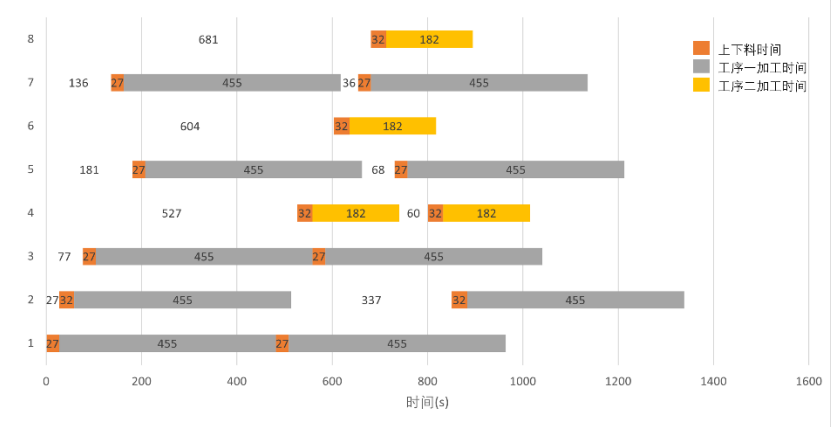

图 5-6 CNC 状态变化图（情况二）

其中，两道工序加工的第一组结果（详见 EXCEL 表）如下：

表 5-7 两道工序调度模型第一组数据部分结果表

加工物料序号	工序 1 的 CNC 编号	上料开始时间	下料开始时间	工序 2 的 CNC 编号	上料开始时间	下料开始时间
1	2	0	431	1	462	918
2	4	51	510	3	541	1022
3	8	115	597	5	628	1126
4	6	166	676	7	732	1230

两道工序加工第二组结果（详见 EXCEL 表）如下：

表 5-8 两道工序调度模型第二组数据部分结果表

加工物料序号	工序 1 的 CNC 编号	上料开始时间	下料开始时间	工序 2 的 CNC 编号	上料开始时间	下料开始时间
1	2	0	315	1	350	901
2	4	58	403	3	438	1037
3	6	116	491	5	526	1155
4	2	315	630	1	665	1273

两道工序加工第三组结果（详见 EXCEL 表）如下：

表 5-9 两道工序调度模型第三组数据部分结果表

加工物料序号	工序 1 的 CNC 编号	上料开始时间	下料开始时间	工序 2 的 CNC 编号	上料开始时间	下料开始时间
1	1	0	482	4	527	741
2	2	27	559	6	604	880
3	3	77	654	4	741	982
4	7	136	791	6	880	1116

5.2.6. 检验分析

在上文中，针对加工用时比的非整数性，对两道工序的用时均采用跨度 100 的四舍五入法，将浮点数转化为整数后再进行比例约分。

下面针对这一问题，采用高斯函数，向下取整：

表 5-10 第一组模型检验结果

	原配比	检验配比
刀具配比	4:4	5:3
平均作业效率	0.008371	0.006909

表 5-11 第二组模型检验结果

	原配比	检验配比
刀具配比	3:5	2:6
平均作业效率	0.007641	0.005886

表 5-12 第三组模型检验结果

	原配比	检验配比
刀具配比	6:2	5:3
平均作业效率	0.007821	0.008052

比较检验结果可知，第三组采用 5:3 的刀具配比时，与 6:2 的刀具配比相较而言，系统的平均作业效率更高，故使用第三组采用 5:3 的刀具配比。

另外得出，当对 CNC 刀具配比进行微调时，智能加工系统的产率不发生显著降低，模型效果好，验证了模型的正确性。

5.2.7. 小结

针对物料加工共两道工序且无故障的情况，仍以一轮加工的最小用时和最大产出为目标函数，总加工时长不超过 8h 为约束条件，区别于情况一模型的是：CNC 对两道工序的加工时长是有区别的，故加入逻辑变量判断当前物料所在工序，区分第一道工序与第二道工序的加工时长，于是建立了 0-1 规划模型，解决了两道工序无故障情况下的 RGV 调度问题。

对于模型的求解，利用动态规划的算法思想进行求解，最终得出：第一组参数情况下，RGV 运行顺序：CNC#2→CNC#4→CNC#8→CNC#6→$(CNC\#2\rightarrow CNC\#1\rightarrow CNC\#4\rightarrow CNC\#3\rightarrow CNC\#6\rightarrow CNC\#5\rightarrow CNC\#8\rightarrow CNC\#7)_{循环}$ 最终成料产出量为 253；第二组参数情况下，RGV 运行顺序 CNC#2→CNC#4→CNC#6→CNC#2→$(CNC\#1\rightarrow CNC\#4\rightarrow CNC\#3\rightarrow CNC\#6\rightarrow CNC\#5\rightarrow CNC\#2)_{循环}$：最终成料产出量为 225；第三组参数情况下，RGV 运行顺序：CNC#1→CNC#2→CNC#3→CNC#7→CN#8→CNC#5→CN#1→CN#4→CNC#3→CNC#6→CNC#7→$(CNC\#4\rightarrow CNC\#2\rightarrow CNC\#6\rightarrow CNC\#5\rightarrow CNC\#4\rightarrow CNC\#1\rightarrow CNC\#6\rightarrow CNC\#3)_{循环}$ 最终成料产出量为 231。

5.3. 情况三模型的建立与求解

5.3.1. 基于空闲时间插入法的故障扰动模型

情况三种物料加工作业仅一道工序，但 CNC 可能发生故障，我们同样需要制定策略，使得系统在一个班次内，实现最大成料产出，且在一个运行回路中的耗时尽可能小。

由题可知，CNC 故障的发生概率约为 1%，因此故障间隔时间 x 一般服从离散时间分布-泊松分布[3]，$P(X=k)=\theta=\frac{e^{-\lambda}\lambda^{k}}{k!}, k=0,1,2,...,n$，即 $X\sim\pi(\lambda)$，故 CNC 故障概率为$P_i=\theta p_i$，$p_i=1\%$。

由题可知，每次故障排除（人工处理，未完成的物料报废）时间介于 10~20 分钟之间，即 600s~(600+600)s，则设置故障排除时间为自 600s 至 1200s 之间的一个随机数$R=600+600*rand(1)$。

基于以上，于情况一模型中的系统总耗时公式中插入一项由 CNC 故障导致的对第 n 个物料的额外等待时间$D_n=R*P_i$，建立一道工序下基于空闲时间插入法的故障扰动模型：

$$\max\ n$$

$$\min\ T_i=(T_d+T_l+T_z)_{\min}+T_j+T_q+D_n$$

$$\begin{cases}T_1=0\\ \Sigma T_i\leqslant 8\times 60\times 60\\ i=1,2,...,n\end{cases}\qquad(5)$$

其中，T_i代表一个班次内的系统总耗时，T_d代表 CNC 等待时间，T_l代表 RGV 旅行时间，T_z代表 RGV 上下料作业时间，T_j代表 CNC 加工时间，T_q代表 RGV 清洗作业时间，D_n代表系统对第 n 个物料的额外等待时间。

（1）初始状态：RGV 位于 CNC#1 前，不论先为哪台 CNC 投放物料，初始时间总为 0；

（2）迭代方程：每一次迭代过后，RGV 得到最短时间$\min\ T_i$，迭代 n 次；

（3）最优值方程：系统一个班次的工作时间为 8h，换算时间单位为 s 得到最优值方程。

5.3.2. 求解算法的设计思路

基于情况一中的求解算法，该空闲时间插入的算法步骤为：

Step1：首先确定处于工作状态的 CNC，若 CNC 处于工作状态则进入 Step2.

Step2：在“RGC 位置及状态判断”函数中考虑故障情况，从而影响可供 RGV 决策集.

Step3：在迭代中对 8 个 CNC 设置故障扰动.

Step4：确定出现故障的 CNC 编号.

Step5：将故障处理时间插入更新机制，同时，制止本次循环中故障加工物料对产出量的增加影响，而直接进行下一次迭代.

5.3.3. 情况三的结论

RGV 调度策略：针对第一组参数情况，按照 1→2→3→4→7→8→5→6→1→2→3→7→5→1→2→3→4→6→5→7→8（详见附录）的顺序加工；针对第二组参数情况，按照 1→2→3→4→7→8→5→6→1→2→3→4→5→6→7→8（详见附录）的顺序加工；针对第三组参数情况，按照 CNC#1→2→3→4→7→8→5→6→1→2→3→4→5→6→7→8→1→2→3→4→5→6→7→8（详见附录）的顺序加工。三组情况最终成料产量分别是 373、343、378。

表 5-13 一道工序故障加工最大产量

	第一组	第二组	第三组
最大产量 M（个）	373	343	378

其中，一道工序 CNC 故障的第一组结果如下（详见 EXCEL 表）：

表 5-14 一道工序调度模型第一组数据部分结果表

加工物料序号	加工 CNC 编号	上料开始时间	下料开始时间
1	1	0	588
2	2	28	641
3	3	79	717
4	4	107	803

表 5-15 第一组的故障

故障时的物料序号	故障 CNC 编号	故障开始时间	故障结束时间
166	1	12674	13504
306	8	23512	24115

一道工序 CNC 故障的第二组结果如下（详见 EXCEL 表）：

表 5-16 一道工序调度模型第二组数据部分结果表

加工物料序号	加工 CNC 编号	上料开始时间	下料开始时间
1	1	0	610
2	2	30	670
3	3	88	758
4	4	118	818

表 5-17 第二组的故障

故障时的物料序号	故障 CNC 编号	故障开始时间	故障结束时间
60	4	4941	5549
128	1	10244	11354
143	2	11593	12654
263	6	21481	22387
286	6	23590	24213
327	8	26479	27355

一道工序 CNC 故障的第三组结果如下（详见 EXCEL 表）：

表 5-18 一道工序调度模型第三组数据部分结果表

加工物料序号	加工 CNC 编号	上料开始时间	下料开始时间
1	1	0	572
2	2	27	624
3	3	77	699
4	4	104	751

表 5-19 第三组的故障

故障时的物料序号	故障 CNC 编号	故障开始时间	故障结束时间
100	4	7648	8824
129	3	9910	10913
170	6	12623	13663
268	1	19733	20532
324	3	23915	24846
336	8	25090	25707

5.3.5. 小结

针对物料加工仅一道工序且 CNC 可能发生故障的情况，于情况一模型基础上给系统总耗时公式增加一项对由故障物料所引起的额外等待时间，仍以一轮加工的最小用时和最大产出为目标函数，总加工时长不超过 8h 为约束条件，建立基于空闲时间插入法的故障扰动模型。

针对模型的求解，在情况二算法基础上，增加对 CNC 是否处于工作状态的

判断，同时确定出现故障的 CNC 编号，然后将故障处理时间插入系统总耗时公式中，同时制止本物料对产出量的中增加影响，直接进入下一轮循环，其他部分延用情况二的动态规划算法。

最终得出动态调度策略为：针对第一组参数情况，按照 1→2→3→4→7→8→5→6→1→2→3→7→5→1→2→3→4→6→5→7→8（详见附录）的顺序加工；针对第二组参数情况，按照 1→2→3→4→7→8→5→6→1→2→3→4→5→6→7→8（详见附录）的顺序加工；针对第三组参数情况，按照 CNC#1→2→3→4→7→8→5→6→1→2→3→4→5→6→7→8→1→2→3→4→5→6→7→8（详见附录）的顺序加工。三组情况最终成料产量分别是 373、343、378。

5.4. 情况四模型的建立与求解

5.4.1. 两道工序下基于空闲时间插入的故障扰动模型

情况四中物料加工作业共两道工序，需要考虑 CNC 出现故障的情况，我们同样需要制定策略，使得系统在一个班次内，实现最大成料产出，且在一个运行回路中的耗时尽可能小。

在情况二的模型基础上，我们引入同情况三模型中相同的空闲时间变量 $D_n = R^*P_i$，$R = 600 + 600^*rand(1)$，$P_i = \theta p_i$，$p_i = 1\%$，

其中 $P(X=k) = \theta = \frac{e^{-\lambda}\lambda^k}{k!}, k = 0, 1, 2, ..., n$。于是建立两道工序下基于空闲时间插入的故障扰动模型：

$$\max\ n$$

$$\min\ T_i = (T_d + T_l + T_z) + (1 - C_{ij})T_{j1} + C_{ij}T_{j2} + C_{ij}T_q + D_n$$

$$\begin{cases} T_1 = 0 \\ \Sigma T_i \leqslant 8 \times 60 \times 60 \\ C_{ij} = 0 \text{或} 1 \\ i,j = 1,2,...,n \end{cases} \tag{6}$$

其中，对于逻辑变量$C_{ij}(i,j = 1,2,...,n)$，有

$$C_{ij} = \begin{cases} 0，\text{第}i\text{次迭代的第}j\text{个物料处在第一道工序} \\ 1，\text{第}i\text{次迭代的第}j\text{个物料处在第二道工序} \end{cases}$$

T_i代表第i次迭代的系统总耗时，T_d代表 CNC 等待时间，T_l代表 RGV 旅行时间，T_z代表 RGV 上下料作业时间，T_j代表 CNC 加工时间，T_q代表 RGV 清洗作业时间，C_{ij}代表第i次迭代中的第j个物料所在的工序。

（1）初始状态：RGV 位于 CNC#1 前，不论先为哪台 CNC 投放物料，初始时间总为 0；

（2）迭代方程：每一次迭代过后，RGV 得到最短时间$\min\ T_i$，迭代 n 次；

（3）最优值方程：系统一个班次的工作时间为 8h，换算时间单位为 s 得到最优值方程。

5.4.2. 求解算法的设计思路

基于情况二中的求解算法，该空闲时间插入的算法步骤为：

Step1：首先确定处于工作状态的 CNC，若 CNC 处于工作状态则进入 Step2.

Step2：在“RGC 位置及状态判断”函数中考虑故障情况，从而影响可供 RGV 决策集.

Step3：在迭代中对 8 个 CNC 设置故障扰动.

Step4：确定出现故障的 CNC 编号.

Step5：将故障处理时间插入更新机制，同时，制止本次循环中故障加工物料对产出量的增加影响，而直接进行下一次迭代.

5.4.3. 情况四的结论

RGV 调度策略：针对第一组参数情况，按照 2→4→8→6→2→1→4→3→6→5→8→7→2→1（详见附录）的顺序加工；针对第二组参数情况，按照 2→4→6→2→1→4→3→6→5（详见附录）的顺序加工；针对第三组参数情况，按照 1→2→3→5→7→1→4→3→6→5→8→7→4→2→6（详见附录）的顺序加工。三组情况最终成料产量分别是 237、218、231。

根据 CNC 加工完成一个两道工序的物料过程中第一和第二道工序所需的时间，得到下表的分配比例。

表 5-20 每组数据的刀具分配比例

	第一组	第二组	第三组
刀具分配比例	4:4	3:5	5:3

首先将题目给出的三组参数分别带入模型，利用 MATLAB 编程求解。经过配比检验，取其中最优的运行方案，得到第一组参数下确定使用 4:4 的放置方案，得到成品 237 个；第二组参数下确定使用 3:5 的模型中求解，得到成品 218 个；第三组 6:2 得到成品 231 个。

其中，两道工序加工故障的第一组结果（详见 EXCEL 表）如下：

表 5-21 两道工序故障调度模型第一组数据部分结果表

加工物料序号	工序 1 的 CNC 编号	上料开始时间	下料开始时间	工序 2 的 CNC 编号	上料开始时间	下料开始时间
1	2	0	431	1	462	918
2	4	51	510	3	541	1022
3	8	115	597	5	628	1126
4	6	166	676	7	707	1230

表 5-22 第一组的故障

故障时的物料序号	故障 CNC 编号	故障开始时间	故障结束时间
27	8	2008	2617
33	6	2063	3185
186	7	11221	11914
378	1	22540	23231
457	8	27047	27848
485	8	28938	29956

两道工序加工故障第二组部分结果（详见 EXEL 表）如下：

表 5-23 两道工序故障调度模型第二组数据部分结果表

加工物料序号	工序 1 的 CNC 编号	上料开始时间	下料开始时间	工序 2 的 CNC 编号	上料开始时间	下料开始时间
1	2	0	315	1	350	753
2	4	58	403	3	438	871
3	6	116	491	5	526	1010
4	2	315	630	1	665	1128

表 5-24 第二组的故障

故障时的物料序号	故障 CNC 编号	故障开始时间	故障结束时间
36	4	2290	3449
65	1	4409	5495
77	7	5584	6632
149	1	9763	10539
223	5	14467	15563
262	2	16923	17683

两道工序加工故障第三组部分结果（详见 EXCEL 表）如下：

表 5-25 两道工序故障调度模型第三组数据部分结果表

加工物料序号	工序 1 的 CNC 编号	上料开始时间	下料开始时间	工序 2 的 CNC 编号	上料开始时间	下料开始时间
1	1	0	482	4	527	797
2	2	27	559	6	604	936
3	3	77	636	8	681	1070
4	5	122	713	4	797	1172

表 5-26 第三组的故障

故障时的物料序号	故障 CNC 编号	故障开始时间	故障结束时间
45	4	2774	3763
192	5	11742	12594
276	1	17197	18130

5.4.4. 系统作业效率

第i组参数情况下的产率η_i定义如下：

$$\eta_i = \frac{M_{2i}}{T_{2i}} \tag{7}$$

其中M_i表示情况 2 中第i组参数情况下的最终产量，T_i表示情况 2 中第i组参数情况下在 8h 周期内最后一件成料完工的系统总耗时。

通过计算可得：

$\overline{\eta_1} = 0.008371448$，$\overline{\eta_2} = 0.007641062$，$\overline{\eta_3} = 0.007821088$.

5.4.5. 检验分析

在上文中，针对加工用时比的非整数性，对两道工序的用时均采用跨度 100 的四舍五入法，将浮点数转化为整数后再进行比例约分。

下面针对这一问题，采用高斯函数，向下取整：

表 5-27 第一组模型检验结果

	原配比	检验配比
刀具配比	4:4	5:3
平均作业效率	0.008767	0.007039

表 5-28 第二组模型检验结果

	原配比	检验配比
刀具配比	3:5	2:6
平均作业效率	0.007846	0.006074

表 5-29 第三组模型检验结果

	原配比	检验配比
刀具配比	5:3	6:2
平均作业效率	0.008074	0.008044

比较检验结果可知，当对 CNC 刀具配比进行微调时，智能加工系统的产率不发生显著降低，模型效果好，验证了模型的正确性。

5.4.6. 小结

针对物料加工两道工序且 CNC 可能发生故障的情况，于情况二模型基础上给系统总耗时公式增加一项对由故障物料所引起的额外等待时间，仍以一轮加工的最小用时和最大产出为目标函数，总加工时长不超过 8h 为约束条件，建立基于空闲时间插入法的故障扰动模型。

针对模型的求解，以情况二与情况三算法为基础，判断 CNC 工作状态以及当前物料所在工序，其他部分延用情况二、三的动态规划算法。

得出动态调度策略为：针对第一组参数情况，按照 2→4→8→6→2→1→4→3→6→5→8→7→2→1（详见附录）的顺序加工；针对第二组参数情况，按照 2→4→6→2→1→4→3→6→5（详见附录）的顺序加工；针对第三组参数情况，按照 1→2→3→5→7→1→4→3→6→5→8→7→4→2→6（详见附录）的顺序加工。三组情况最终成料产量分别是 237、218、231。

最后，通过更换刀具配比这一模型参数的计算方法，进而修改了参数值，再次计算系统平均作业效率，得知模型较为有效，实现了对系统作业效率和模型有效性的检验。

六、模型的评价、改进与推广

6.1. 模型的评价

1. 模型的优点

（1）建立规划模型和求解，得出了 RGV 动态调度的有效策略；

（2）使用动态规划使得模型的求解具有灵活性，避免了重复计算，降低了时间复杂度。

2. 模型的缺点

（1）0-1 规划模型求解困难；

（2）使用动态规划使得空间复杂度较高。

6. 2. 模型的改进

(1) 可以进一步深入研究系统状态的特性，例如：周期性、波动性等。通过对特性进行概括和证明，并据此改进优化过程，有助于进一步提升模型的求解精度和求解速度。

(2) 可以使用遗传算法对原始模型进行求解。遗传算法借鉴了自然选择和进化机制，是一种高度平行、随机、自适应搜索算法[4]，适合解决本题的双目标优化模型。需要注意的是，如果从时间和状态转移视角会由于时间划分过细导致染色体基因数目过多问题，或者基于事件编码时，由于各个事件的时间不确定，不便于对所选方案进行编码。所以采用可以采用可变长染色体的遗传算法（Messy GA）[5]。此外在染色体交叉、变异操作时应当保证操作后的染色体仍在可行解区域内（不能出现一个物料被两个 CNC 同时加工），因此需要限制变异和交叉操作的有效范围。从而提升算法求解的可靠性。其基本模型如下：

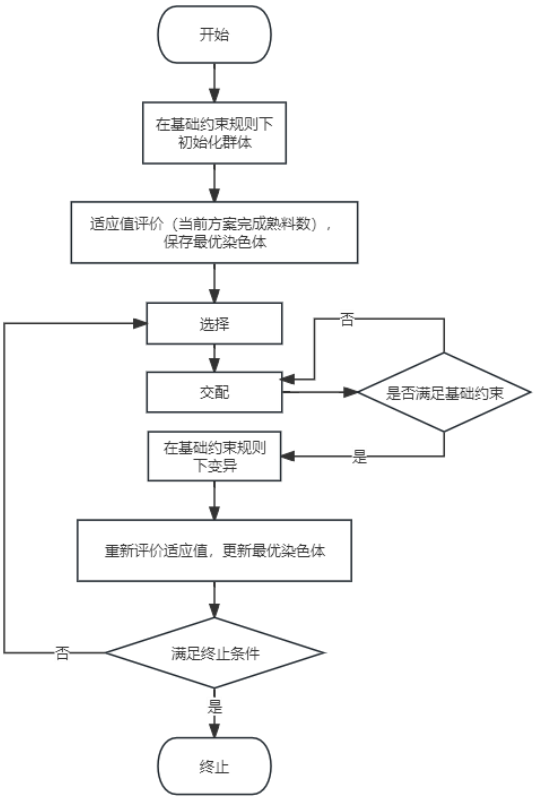

图 5-7 改进遗传算法流程图

参考文献

[1]Fu J, Zhang H, Zhang J, et al. Review on AGV scheduling optimization[J]. Journal of System Simulation, 2020, 32(9): 1664.

[2]Lu S, Pei J, Liu X, et al. A hybrid DBHVNS for highend equipment production scheduling with machine failures and preventive maintenance activities[J]. Journal of Computational and Applied Mathematics, 2021, 384: 113195.

[3]刘琳.动态不确定环境下生产调度算法研究[D].上海:上海交通大学，2007.

[4]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.

[5]Hettiarachchi D S , Noman N , Iba H .Messy Genetic Algorithm for evolving mathematical function evaluating variable length gene regulatory networks[C]//Evolutionary Computation.IEEE, 2013.DOI:10.1109/CEC.2013.6557824.

附录

附录 1
介绍：一道工序故障情况第 2 组最优路径
[1;2;3;4;7;8;5;6;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;5;6;7;8;4;1;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;2;3;5;6;7;8;4;2;1;3;5;7;8;4;1;3;5;6;7;8;4;1;2;3;5;7;8;4;1;2;3;5;6;7;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7;8;4;1;2;3;5;6;7]

附录 2
介绍：一道工序故障情况第 3 组最优路径
[1;2;3;4;7;8;5;6;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;5;6;7;8;1;2;3;5;6;7;8;1;2;3;4;5;6;7;8;1;2;3;4;5;6;7;8;1;2;4;5;6;7;8;1;2;4;5;6;7;8;1;3;2;4;5;6;7;8;1;3;2;4;5;6;7;8;1;3;2;4;5;6;7;8;1;3;2;4;5;7;8;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;2;4;5;7;8;6;1;3;2;4;

5;7;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6;1;3;2;4;5;7;8;6]

附录 3

介绍：两道工序故障情况第 1 组最优路径

[2;4;8;6;2;1;4;3;6;5;8;7;2;1;4;3;6;5;8;7;2;1;4;3;6;5;8;7;2;1;4;3;6;5;2;1;4;3;2;1;4;3;8;7;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;7;6;5;2;1;4;3;8;5;6;1;2;3;4;5;8;1;2;3;4;7;6;5;8;3;2;7;6;5;4;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7;6;5;4;1;2;3;6;5;4;1;2;3;6;5;4;1;2;3;8;7;6;5;4;1;2;3;8;7]

附录 4

介绍：两道工序故障情况第 2 组最优路径

[2;4;6;2;1;4;3;6;5;2;1;4;3;6;5;2;1;4;3;6;5;2;1;4;3;6;5;2;1;4;3;6;5;2;1;4;3;6;5;2;1;6;5;2;1;6;5;2;1;6;5;2;1;4;3;6;5;2;1;4;3;6;5;2;1;4;3;6;5;2;7;6;5;4;3;2;7;6;5;4;3;2;8;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;7;6;5;4;3;2;7;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;7;4;3;2;1;6;7;4;3;2;1;6;7;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;6;5;4;3;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;4;3;2;1;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;7;6;5;4;3;2;7;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6;5;4;3;2;1;6]

附录 5

介绍：两道工序故障情况第 3 组最优路径

[1;2;3;5;7;1;4;3;6;5;8;7;4;2;6;1;4;3;6;5;8;7;6;2;4;1;6;5;4;3;6;7;8;2;4;1;6;5;4;3;6;7;8;2;4;1;6;5;8;7;6;3;8;1;6;5;8;7;6;3;8;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;

2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;5;4;2;6;3;4;1;6;7;8;2;4;3;6;1;4;7;8;5;6;3;4;2;6;1;4;5;6;7;8;3;4;2;6;1;4;5;6;7;8;3;4;2;6;1;4;5;6;7;8;3;4;2;6;1;4;5;6;7;8;3;4;2;6;1;4;5;6;7;8;3;4;2;6;1;4;5;6;7;8;3;4;2;6;1;4;5;6;7;8;3;4;2;6;5;4;7;8;3;4;2;6;5;4;7;8;3;4;2;6;5;4;1;6;3]

附录 6

介绍：模型一求解代码

```
function main_1()
% Initialize variables
    totalTime = 0;
    maxOutput = 0;
    optimalPath = [];
    RGVPosition = 1; % Assuming RGV starts at position 1
    CNCProcessingTime = zeros(1, 8); % Time required for each CNC to
process a workpiece
    CNCWorkStatus = zeros(1, 8); % 1 if CNC is processing, 0 if not
    %materialDemandPoints = [1,2,3,4,5,6,7,8]; % List of material demand
points (CNC machine locations)
    upload = [];

    Tz = [27,32];%[28,31];[30,35];[27,32];
    Tq = 25;%25;30;25;

    while totalTime <= 28800
% Step 1: Calculate the shortest time required to move to the next
material demand point
        [shortestTime,nextCNC] = calculateShortestTime(RGVPosition,
CNCWorkStatus, CNCProcessingTime);

% Step 2: Move RGV to the next material demand point
        moveTime = shortestTime;
        totalTime = totalTime + moveTime;
        RGVPosition = nextCNC;

        [CNCProcessingTime, ~] = Elapsing(CNCProcessingTime,
CNCWorkStatus, moveTime);
        upload = [upload; totalTime];
% Step 3: Perform RGV material loading/unloading operations and update
CNC status
        totalTime = totalTime + Tz(2-mod(RGVPosition,2));

        [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCProcessingTime,
CNCWorkStatus, Tz(2-mod(RGVPosition,2)));

        CNCProcessingTime(RGVPosition) = eps;
        CNCWorkStatus(RGVPosition) = 1;

% Step 4: Clean the material and complete a cycle
        maxOutput = maxOutput + 1;
```

```
        optimalPath = [optimalPath; RGVPosition];
        if maxOutput >= 9
            totalTime = totalTime + Tq;
            [CNCProcessingTime, CNCWorkStatus] =
Elapsing(CNCProcessingTime, CNCWorkStatus, Tq);
        end


% Step 5: Update total time and check if it exceeds 8 hours
        if totalTime <= 28800
            % Update the max output and optimal path if needed
        else
            break; % Exit the loop if the total time exceeds 8 hours
        end
    end

% Output the results
    fprintf('Maximum material output: %d\n', maxOutput-9);
    fprintf('totalTime: %d\n', totalTime);
    fprintf('Optimal path: %s\n', mat2str(optimalPath));
    fprintf('upload: %s\n', mat2str(upload));
end



function [shortestTime, nextCNC] = calculateShortestTime(RGVPosition,
CNCWorkStatus, CNCProcessingTime)
    Tj = 545;%560;580;545;
    Tz = [27,32];%[28,31];[30,35];[27,32];
    Time = zeros(1,8);
    for n = 1:8
        if CNCWorkStatus(n) == 0
            Time(n) = calculateMovingTime(RGVPosition,n) + Tz(2-mod(n,2));
        else
            Time1 = calculateMovingTime(RGVPosition,n) + Tz(2-mod(n,2));
            Time2 = Tj-CNCProcessingTime(n) + Tz(2-mod(n,2));
            Time(n) = max(Time1, Time2);
        end
    end
    [shortestTime, nextCNC] = min(Time);
    shortestTime = shortestTime - Tz(2-mod(nextCNC,2));
end


function [MovingTime] = calculateMovingTime(RGVPosition,n)
    if(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-1))==0)
        MovingTime=0;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==1)
        MovingTime=18;%20;23;18;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==2)
        MovingTime=32;%33;41;32;
    else,MovingTime=46;%46;59;46;
    end
end
```

```
function [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCProcessingTime,
CNCWorkStatus, T)
    Tj = 545;%560;580;545;
    CNCProcessingTime = CNCProcessingTime + T .* CNCWorkStatus;
    for i = 1:8
        if CNCProcessingTime(i) >= Tj
            CNCProcessingTime(i) = 0;
            CNCWorkStatus(i) = 0;
        end
    end
    end
```

附录 7

介绍：模型二求解代码

```
function main_2()
% Initialize variables

    C = 0;
    totalTime = 0;
    maxOutput = 0;
    optimalPath = [];
    RGVPosition = 1; % Assuming RGV starts at position 1
    CNCProcessingTime = zeros(1, 8); % Time required for each CNC to
process a workpiece
    CNCWorkStatus = zeros(1, 8); % 1 if CNC is processing, 0 if not
    %materialDemandPoints = [1,2,3,4,5,6,7,8]; % List of material demand
points (CNC machine locations)
    upload1 = [];
    upload2 = [];
    download1 = [];
    download2 = [];
    k = 0;

    Tz = [27,32];%[28,31];[30,35];[27,32];
    Tq = 25;%25;30;25;

    while totalTime <= 28800
        k = k + 1;

% Step 1: Calculate the shortest time required to move to the next
material demand point
        [shortestTime,nextCNC] = calculateShortestTime(C, k, RGVPosition,
CNCWorkStatus, CNCProcessingTime);

% Step 2: Move RGV to the next material demand point
        moveTime = shortestTime;
        totalTime = totalTime + moveTime;
        RGVPosition = nextCNC;

        [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCProcessingTime,
CNCWorkStatus, moveTime);

        if C == 0, upload1 = [upload1; totalTime];end
        if C == 1, upload2 = [upload2; totalTime];end
        if C == 0 && k >=6, download1 = [download1; totalTime];    end
        if C == 1 && k >=11, download2 = [download2; totalTime];    end
```

```
% Step 3: Perform RGV material loading/unloading operations and update
CNC status
        totalTime = totalTime + Tz(2-mod(RGVPosition,2));

        [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCProcessingTime,
CNCWorkStatus, Tz(2-mod(RGVPosition,2)));

        CNCProcessingTime(RGVPosition) = eps;
        CNCWorkStatus(RGVPosition) = 1;
        if k >= 6, C = 1 - C; end

% Step 4: Clean the material and complete a cycle
        optimalPath = [optimalPath; RGVPosition];
        if C == 1 && k >= 11
            maxOutput = maxOutput + 1;
            totalTime = totalTime + Tq;
            [CNCProcessingTime, CNCWorkStatus] =
Elapsing(CNCProcessingTime, CNCWorkStatus, Tq);
        end


% Step 5: Update total time and check if it exceeds 8 hours
        if totalTime <= 28800
            % Update the max output and optimal path if needed
        else
            break; % Exit the loop if the total time exceeds 8 hours
        end
    end

% Output the results
    fprintf('Maximum material output: %d\n', maxOutput);
    fprintf('totalTime: %d\n', totalTime);
    fprintf('Optimal path: %s\n', mat2str(optimalPath));
    fprintf('upload1: %s\n', mat2str(upload1));
    fprintf('download1: %s\n', mat2str(download1));
    fprintf('upload2: %s\n', mat2str(upload2));
    fprintf('download2: %s\n', mat2str(download2));
end


function [shortestTime, nextCNC] = calculateShortestTime(C, k,
RGVPosition, CNCWorkStatus, CNCProcessingTime)
    Tj1 = 455;%400;280;455;
    Tj2 = 182;%378;500;182;
    I = [1,2,3,5,7];
    J = [4,6,8];
    Time1 = zeros();
    Time2 = zeros();
    Time = [inf,inf,inf,inf,inf,inf,inf,inf];
   if k == 1, nextCNC = 1; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 2, nextCNC = 2; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 3, nextCNC = 3; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 4, nextCNC = 7; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
```

```
   if k == 5, nextCNC = 5; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 6, nextCNC = 1; shortestTime = Tj1-CNCProcessingTime(1);end
   if k > 6
       if C == 0
           for i = 1:5
           if CNCWorkStatus(I(i)) == 0
               Time(I(i)) = calculateMovingTime(RGVPosition,I(i));
           else
               Time1(I(i)) = calculateMovingTime(RGVPosition,I(i));
               Time2(I(i)) = Tj1-CNCProcessingTime(I(i));
               Time(I(i)) = max(Time1(I(i)), Time2(I(i)));
           end
           end
           [shortestTime, nextCNC] = min(Time);

       else
           for j = 1:3
           if CNCWorkStatus(J(j)) == 0
               Time(J(j)) = calculateMovingTime(RGVPosition,J(j));
           else
               Time1(J(j)) = calculateMovingTime(RGVPosition,J(j));
               Time2(J(j)) = Tj2-CNCProcessingTime(J(j));
               Time(J(j)) = max(Time1(J(j)), Time2(J(j)));
           end
           end
           [shortestTime, nextCNC] = min(Time);

       end

   end
end


function [MovingTime] = calculateMovingTime(RGVPosition,n)
    if(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-1))==0)
        MovingTime=0;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==1)
        MovingTime=18;%20;23;18;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==2)
        MovingTime=32;%33;41;32;
    else,MovingTime=46;%46;59;46;
    end
end


function [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCProcessingTime,
CNCWorkStatus, T)
    Tj1 = 455;%400;280;455;
    Tj2 = 182;%378;500;182;

    CNCProcessingTime = CNCProcessingTime + T.* CNCWorkStatus;

for i = 1:8
    if i == 1||2||3||5||7
           if CNCProcessingTime(i) >= Tj1
           CNCProcessingTime(i) = 0;
```

```
            CNCWorkStatus(i) = 0;
            end
    else
            if CNCProcessingTime(i) >= Tj2
            CNCProcessingTime(i) = 0;
            CNCWorkStatus(i) = 0;
            end
    end
end
    end
```

附录 8

介绍：模型三求解代码

```
function main3_1()
% Initialize variables
    totalTime = 0;
    maxOutput = 0;
    optimalPath = [];
    RGVPosition = 1; % Assuming RGV starts at position 1
    CNCProcessingTime = zeros(1, 8); % Time required for each CNC to
process a workpiece
    CNCRepairedTime = zeros(1, 8);
    CNCWorkStatus = zeros(1, 8); % 1 if CNC is processing, 0 if not
    CNCStatus = zeros(1, 8);
    %materialDemandPoints = [1,2,3,4,5,6,7,8]; % List of material demand
points (CNC machine locations)
    upload = [];
    download = [];
    Error = [];

    Tz = [28,31];%[28,31];[30,35];[27,32];
    Tq = 25;%25;30;25;
    k = 0;
    error = 0;

    while totalTime <= 28800
        k = k + 1;
% Step 1: Calculate the shortest time required to move to the next
material demand point

        [shortestTime,nextCNC, CNCWorkStatus, CNCStatus,
CNCProcessingTime, CNCRepairedTime] = calculateShortestTime(k,
RGVPosition, CNCWorkStatus, CNCStatus, CNCProcessingTime,
CNCRepairedTime, totalTime);

% Step 2: Move RGV to the next material demand point
        moveTime = shortestTime;
        totalTime = totalTime + moveTime;
        RGVPosition = nextCNC;

        [CNCProcessingTime, ~] = Elapsing(CNCStatus, CNCProcessingTime,
CNCWorkStatus, moveTime);
        upload = [upload; totalTime];
% Step 3: Perform RGV material loading/unloading operations and update
CNC status
        totalTime = totalTime + Tz(2-mod(RGVPosition,2));
```

```
        [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, Tz(2-mod(RGVPosition,2)));

        CNCProcessingTime(RGVPosition) = eps;
        CNCWorkStatus(RGVPosition) = 1;


        e = rand(1);    if e >= 0.99
                           [Ts,Te] = ErrorElapsing(totalTime);
                           CNCRepairedTime(RGVPosition) = Te;
                           CNCWorkStatus(RGVPosition) = 0;
                           CNCStatus(RGVPosition) = 1;
                           CNCProcessingTime(RGVPosition) = inf;
                           Error = [Error;[k, RGVPosition, Ts, Te]];
                           error = error + 1;
                        end


% Step 4: Clean the material and complete a cycle
        maxOutput = maxOutput + 1;
        optimalPath = [optimalPath; RGVPosition];
        if k >= 9
            totalTime = totalTime + Tq;
            [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, Tq);
        end


% Step 5: Update total time and check if it exceeds 8 hours
        if totalTime <= 28800
            % Update the max output and optimal path if needed
        else
            break; % Exit the loop if the total time exceeds 8 hours
        end
    end

% Output the results
    fprintf('Maximum material output: %d\n', maxOutput-9-error);
    fprintf('totalTime: %d\n', totalTime);
    fprintf('Optimal path: %s\n', mat2str(optimalPath));
    fprintf('upload: %s\n', mat2str(upload));
    fprintf('Error: %s\n', mat2str(Error));
end




function [shortestTime, nextCNC, CNCWorkStatus, CNCStatus,
CNCProcessingTime, CNCRepairedTime] = calculateShortestTime(k,
RGVPosition, CNCWorkStatus, CNCStatus, CNCProcessingTime,
CNCRepairedTime, totalTime)
    Tj = 560;%560;580;545;
    Tz = [28,31];%[28,31];[30,35];[27,32];
    Time = zeros(1,8);
```

```
   if k == 1, nextCNC = 1; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 2, nextCNC = 2; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 3, nextCNC = 3; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 4, nextCNC = 4; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 5, nextCNC = 7; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 6, nextCNC = 8; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 7, nextCNC = 5; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 8, nextCNC = 6; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k > 8

    for n = 1:8
        if CNCStatus(n) == 0
            if CNCWorkStatus(n) == 0
            Time(n) = calculateMovingTime(RGVPosition,n) + Tz(2-mod(n,2));
            else
            Time1 = calculateMovingTime(RGVPosition,n) + Tz(2-mod(n,2));
            Time2 = Tj-CNCProcessingTime(n) + Tz(2-mod(n,2));
            Time(n) = max(Time1, Time2);
            end
        else
            if totalTime >= CNCRepairedTime(n)
            Time(n) = calculateMovingTime(RGVPosition,n) + Tz(2-mod(n,2));
CNCWorkStatus(n) = 0;    CNCStatus(n) = 0;    CNCProcessingTime(n) = 0;
CNCRepairedTime(n) = 0;
            else
            Time(n) = inf;
            end
        end
    end
    [shortestTime, nextCNC] = min(Time);
    shortestTime = shortestTime - Tz(2-mod(nextCNC,2));
    end
end


function [MovingTime] = calculateMovingTime(RGVPosition,n)
    if(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-1))==0)
        MovingTime=0;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==1)
        MovingTime=20;%20;23;18;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==2)
        MovingTime=33;%33;41;32;
    else,MovingTime=46;%46;59;46;
    end
end


function [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, T)
```

```
    Tj = 560;%560;580;545;
    CNCProcessingTime = CNCProcessingTime + T .* CNCWorkStatus;
    for i = 1:8
        if CNCProcessingTime(i) >= Tj && CNCStatus(i) == 0
            CNCProcessingTime(i) = 0;
            CNCWorkStatus(i) = 0;
        end
    end
end

function [Ts, Te] = ErrorElapsing(T)
    Tj = 560;%560;580;545;
    Ts = T + Tj*rand(1);
    Te = Ts + 600 + 600*rand(1);
end
```

附录 9

介绍：模型四求解代码

```
function main3_2()
% Initialize variables

    C = 0;
    totalTime = 0;
    maxOutput = 0;
    optimalPath = [];
    optimalPath1 = [];
    optimalPath2 = [];
    RGVPosition = 1; % Assuming RGV starts at position 1
    CNCProcessingTime = zeros(1, 8); % Time required for each CNC to
process a workpiece
    CNCRepairedTime = zeros(1, 8);
    CNCWorkStatus = zeros(1, 8); % 1 if CNC is processing, 0 if not
    CNCStatus = zeros(1, 8);
    %materialDemandPoints = [1,2,3,4,5,6,7,8]; % List of material demand
points (CNC machine locations)
    upload1 = [];
    upload2 = [];
    download1 = [];
    download2 = [];
    Error = [];
    error = 0;

    k = 0;

    Tz = [28,31];%[28,31];[30,35];[27,32];
    Tq = 25;%25;30;25;

    while totalTime <= 28800
        k = k + 1;

% Step 1: Calculate the shortest time required to move to the next
material demand point
        [shortestTime,nextCNC, CNCWorkStatus, CNCStatus,
CNCProcessingTime, CNCRepairedTime] = calculateShortestTime(C, k,
RGVPosition, CNCWorkStatus, CNCStatus, CNCProcessingTime,
CNCRepairedTime, totalTime);
```

```
% Step 2: Move RGV to the next material demand point
        moveTime = shortestTime;
        totalTime = totalTime + moveTime;
        RGVPosition = nextCNC;

        [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, moveTime);

        if C == 0, upload1 = [upload1; totalTime];    end
        if C == 1, upload2 = [upload2; totalTime];    end
        if C == 0 && k >=5, download1 = [download1; totalTime];    end
        if C == 1 && k >=13, download2 = [download2; totalTime];    end


% Step 3: Perform RGV material loading/unloading operations and update
CNC status
        totalTime = totalTime + Tz(2-mod(RGVPosition,2));

        [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, Tz(2-mod(RGVPosition,2)));

        CNCProcessingTime(RGVPosition) = eps;
        CNCWorkStatus(RGVPosition) = 1;

            e = rand(1);    if e >= 0.99
                           [Ts,Te] = ErrorElapsing(C, totalTime);
                           CNCRepairedTime(RGVPosition) = Te;
                           CNCWorkStatus(RGVPosition) = 0;
                           CNCStatus(RGVPosition) = 1;
                           CNCProcessingTime(RGVPosition) = inf;
                           Error = [Error;[k, RGVPosition, Ts, Te]];
                           error = error + 1;
                           end
        if C == 0,    optimalPath1 = [optimalPath1; RGVPosition];
        else,         optimalPath2 = [optimalPath2; RGVPosition];
        end
        optimalPath = [optimalPath; RGVPosition];


        if k >= 5, C = 1 - C; end

% Step 4: Clean the material and complete a cycle
        if C == 1 && k >= 13
            maxOutput = maxOutput + 1;
            totalTime = totalTime + Tq;
            [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, Tq);
        end


% Step 5: Update total time and check if it exceeds 8 hours
        if totalTime <= 28800
            % Update the max output and optimal path if needed
        else
            break; % Exit the loop if the total time exceeds 8 hours
        end
    end
```

```
% Output the results
    fprintf('Maximum material output: %d\n', maxOutput);
    fprintf('totalTime: %d\n', totalTime);
    fprintf('Optimal path1: %s\n', mat2str(optimalPath1));
    fprintf('Optimal path2: %s\n', mat2str(optimalPath2));
    fprintf('Optimal path: %s\n', mat2str(optimalPath));
    fprintf('upload1: %s\n', mat2str(upload1));
    fprintf('download1: %s\n', mat2str(download1));
    fprintf('upload2: %s\n', mat2str(upload2));
    fprintf('download2: %s\n', mat2str(download2));
    fprintf('Error: %s\n', mat2str(Error));
end



function [shortestTime, nextCNC, CNCWorkStatus, CNCStatus,
CNCProcessingTime, CNCRepairedTime] = calculateShortestTime(C, k,
RGVPosition, CNCWorkStatus, CNCStatus, CNCProcessingTime,
CNCRepairedTime, totalTime)
    Tj1 = 400;%400;280;455;
    Tj2 = 378;%378;500;182;
    Tz = [28,31];%[28,31];[30,35];[27,32];
    I = [2,4,6,8];
    J = [1,3,5,7];
    Time1 = zeros();
    Time2 = zeros();
    Time = [inf,inf,inf,inf,inf,inf,inf,inf];
   if k == 1, nextCNC = 2; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 2, nextCNC = 4; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 3, nextCNC = 8; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 4, nextCNC = 6; shortestTime =
calculateMovingTime(RGVPosition,nextCNC);end
   if k == 5, nextCNC = 2; shortestTime = Tj1-CNCProcessingTime(2);end
   if k > 5
       if C == 0
           for i = 1:4
           if CNCStatus(I(i)) == 0
               if CNCWorkStatus(I(i)) == 0
               Time(I(i)) = calculateMovingTime(RGVPosition,I(i));
               else
               Time1(I(i)) = calculateMovingTime(RGVPosition,I(i));
               Time2(I(i)) = Tj1-CNCProcessingTime(I(i));
               Time(I(i)) = max(Time1(I(i)), Time2(I(i)));
               end
           else
               if totalTime >= CNCRepairedTime(I(i))
               Time(I(i)) = calculateMovingTime(RGVPosition,I(i)) + Tz(2-
mod(I(i),2));    CNCWorkStatus(I(i)) = 0;    CNCStatus(I(i)) = 0;
CNCProcessingTime(I(i)) = 0;    CNCRepairedTime(I(i)) = 0;
               else
               Time(I(i)) = inf;
               end
           end
           end
           [shortestTime, nextCNC] = min(Time);
```

```
       else
           for j = 1:4
           if CNCStatus(J(j)) == 0
               if CNCWorkStatus(J(j)) == 0
               Time(J(j)) = calculateMovingTime(RGVPosition,J(j));
               else
               Time1(J(j)) = calculateMovingTime(RGVPosition,J(j));
               Time2(J(j)) = Tj2-CNCProcessingTime(J(j));
               Time(J(j)) = max(Time1(J(j)), Time2(J(j)));
               end
           else
               if totalTime >= CNCRepairedTime(J(j))
               Time(J(j)) = calculateMovingTime(RGVPosition,J(j)) + Tz(2-
mod(J(j),2));    CNCWorkStatus(J(j)) = 0;    CNCStatus(J(j)) = 0;
CNCProcessingTime(J(j)) = 0;    CNCRepairedTime(J(j)) = 0;
               else
               Time(J(j)) = inf;
               end
           end
           end
           [shortestTime, nextCNC] = min(Time);

       end

   end
end


function [MovingTime] = calculateMovingTime(RGVPosition,n)
    if(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-1))==0)
        MovingTime=0;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==1)
        MovingTime=20;%20;23;18;
    elseif(abs(((RGVPosition+mod(RGVPosition,2))/2-1)-((n+mod(n,2))/2-
1))==2)
        MovingTime=33;%33;41;32;
    else,MovingTime=46;%46;59;46;
    end
end


function [CNCProcessingTime, CNCWorkStatus] = Elapsing(CNCStatus,
CNCProcessingTime, CNCWorkStatus, T)
    Tj1 = 400;%400;280;455;
    Tj2 = 378;%378;500;182;

    CNCProcessingTime = CNCProcessingTime + T.* CNCWorkStatus;

for i = 1:8
    if i == 2||4||6||8
           if CNCProcessingTime(i) >= Tj1 && CNCStatus(i) == 0
           CNCProcessingTime(i) = 0;
           CNCWorkStatus(i) = 0;
           end
    else
           if CNCProcessingTime(i) >= Tj2 && CNCStatus(i) == 0
           CNCProcessingTime(i) = 0;
           CNCWorkStatus(i) = 0;
```

```
            end
    end
end
end


function [Ts, Te] = ErrorElapsing(C,T)
    Tj1 = 400;%400;280;455;
    Tj2 = 378;%378;500;182;
    Ts = T + (Tj1*(1-C)+Tj2*C)*rand(1);
    Te = Ts + 600 + 600*rand(1);
    end
```