赛区评阅编号（由赛区组委会填写）：

2018 年高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》（以下简称为“竞赛章程和参赛规则”，可从全国大学生数学建模竞赛网站下载）。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上 QQ 群、微信群等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛章程和参赛规则的，如果引用别人的成果或资料（包括网上资料），必须按照规定的参考文献的表述方式列出，并在正文引用处予以标注。在网上交流和下载他人的论文是严重违规违纪行为。

我们以中国大学生名誉和诚信郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权全国大学生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号（从 A/B/C/D 中选择一项填写）：_____

我们的报名参赛队号（12 位数字全国统一编号）：_____

参赛学校（完整的学校全称，不含院系名）：_____

参赛队员 (打印并签名) ：1._____

2._____

3._____

指导教师或指导教师组负责人　(打印并签名)：_____

（指导教师签名意味着对参赛队的行为和论文的真实性负责）

日期：_____年____月____日

（请勿改动此页内容和格式。此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

赛区评阅编号（由赛区组委会填写）：

2018 年高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号（赛区组委会填写）：

全国评阅随机编号（全国组委会填写）：

（请勿改动此页内容和格式。此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页。）

RGV 智能调度系统模型的研究

摘要

轨道式自动引导车 RGV 能够在生产线导轨上进行无人监督的往返作业，通过 RGV 建立产品生产的智能调度系统能够减少往日人们需进行的许多重复的简单作业，减轻工人负担，提高生产作业效率。为了保证智能调度系统完整运行，且尽可能提高其工作效率，需根据 RGV 工作原理，建立智能调度系统的运转模型。

针对任务（1），我们根据作业流程将 RGV 的工作环节分为初始状态的部署工作和智能调度环节两部分。我们为 CNC 工作台的不同工作状态定义了不同的需求信号，RGV 会根据这些需求信号执行相应操作。当 RGV 接收到来自多个 CNC 工作台的需求信号时，我们为 RGV 定义了两种的优先级选择规则 DOM（Distance-Odd-Minimum）和 DOC（Distance-Odd-Central），求解模型后得出，智能调度系统在不同的优先级选择规则下所得到的求解结果相同。

对于不同工序的生产流程，我们分别讨论了 RGV 的智能调度方案。且对于双工序的生产情况，我们考虑了负责不同工序 CNC 工作台的最优配置数量问题和最优配置位置问题。对于 CNC 的故障模型，我们在程序中设立了故障触发器 Counter 以模拟每个 CNC 工作台 1%的故障发生率。

针对任务（2），我们对 3 组系统参数进行了实际模拟，再次验证不同的优先级选择方案 DOM 和 DOC 对系统的作业效率没有影响。我们定义系统的作业效率为每小时完成的成品件数，单工序无故障条件下，3 组参数下的作业效率分别为 44.625，42.125 和 45.875；双工序无故障条件下，作业效率分别为 29.5，25.25 和 30.125。通过研究发现，对于第 1 组数据，最优 CNC 配置方案需要 4 台 CNC 负责第一道工序，4 台 CNC 负责第二道工序；对于第 2 组数据，上述比例仍为 4:4，对于第三组，比例为 5:3。

此外，我们发现从单工序转变为双工序，第 1 组和第 3 组参数下系统的作业效率降低 33%左右，而第 2 组参数下系统的作业效率将降低 40%以上。故障的发生会使第 1 组和第 3 组的作业效率降低 1%至 2%，而使第 2 组的作业效率降低 3%至 4%。对比表 1 中 3 组不同的参数得出，RGV 在移动，上下料和清洗作业的时间开销会影响系统的工作效率，且开销越大，系统的工作效率越低。因此，为了提高该智能调度系统的作业效率，尽可能提高 RGV 的操作速度，减少 CNC 加工外的时间开销是关键之一。

关键词：DOM 和 DOC 优先级，负责不同工序的 CNC 最优配置方案，作业效率因素

1. 问题重述

背景： 智能加工系统能自动完成产品在生产线生产工序的调度工作。轨道式自动引导车 RGV 能根据人们设定的程序，响应计算机数控机床 CNC 发出的需求信号，并在轨道上往返操作，完成产品生产线的上料，下料，清洗等工作。了解智能加工系统的原理及系统中各组件的功能，我们需完成下面两个任务：

（1） 研究一般情况下的 RGV 动态调度模型和相应的求解算法

（2） 根据表格中所给 3 组系统参数，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

2. 问题分析

任务（1）分析：

从单件产品的生产工序分析，每件产品需经过上料传送带、上料、加工、下料、清洗和下料传送带六个工序；从 RGV 的工作状态分析，RGV 共有移动，暂停等待命令、上下料和清洗四种工作状态；从 CNC 的工作状态分析，CNC 共有等待上料，加工、等待下料和故障四种状态。

从工序逻辑分析，每次 RGV 完成下料工作取下熟料的同时，会执行生料的上料工作，完成该步骤 RGV 应立即让取下的熟料与清洗槽中的成品交换位置完成清洗工作，同时成料也将被放入到下料传动带，完成一道完整工序，之后 RGV 再到下一个 CNC 处，完成生料上料和熟料的下料工作，重复进行上述流程。

在所有 CNC 均在执行加工任务时，RGV 不会收到需求型号故在原地等待；而当有多个 CNC 发出工作需求信号时，应给 RGV 设定任务执行优先级，优先级的选择应该尽可能提高智能控制系统的工作效率。

对于具有需要进行两次加工的产品，由于两次工序所花费的时间不同，分配 CNC 到不同工序的个数需根据两次工序花费的时间决定，需寻求最优的配置方案以提供系统的工作效率。

任务（2）分析：

任务（2）中提供了三组不同的系统配置参数，包括 RGV 移动时间、CNC 不同工序的加工时间、RGV 上下料时间和清洗时间等。按照任务（1）中求解的模型，我们应尽可能在规定时间内能完成的更多的产品，使智能系统的工作效率最高。

3. 模型假设与符号系统

3.1 模型假设

（1）假设 RGV 在相对的 CNC 间转向操作的时间忽略不计

（2）假设生料在上料传送带上的传送时间忽略不计

（3）假设 RGV 在未收到 CNC 需求信号时，只能保持原地待命

（4）假设 RGV 在接收命令和 CNC 需求信号后作出反应的时间忽略不计

3.2 符号系统

符号系统

符号	含义
d_i	RGV 与编号为i的 CNC 工作台的距离（$i = 1,2,\cdots,8$）
C_n^i	编号为n的 CNC 工作台的第i种需求信号（$n = 1,\cdots,8$，$i = 1,\cdots 4$）
M_i	RGV 移动的距离值
T_w^i	RGV 移动i个单位长度花费的时间（$i = 1,2,3$）
L_i	RGV 为 CNC 进行上下料操作的类型（$i = 0,1,2$）
F_N^0	序号为N的产品第一道工序的上料开始时间
F_N^1	序号为N的产品第一道工序的下料开始时间
S_N^0	序号为N的产品第二道工序的上料开始时间
S_N^1	序号为N的产品第二道工序的下料开始时间
T_n	编号为n的 CNC 工作台的故障停机维修用时
T_0	RGV 为编号为偶数的 CNC 工作台上下料用时
T_1	RGV 为编号为奇数的 CNC 工作台上下料用时
t_n^l	RGV 为编号为n的 CNC 工作台上下料用时
t_0	RGV 完成清洗工作所需的时间
t_1	CNC 完成第一道工序加工所需的时间
t_2	CNC 完成第二道工序加工所需的时间
N_0	在规定时间内完成的总产品件数
C_n^{mod}	编号为n的 CNC 工作台的工序状态，取值$0-1$
N_1	参与第一道工序的 CNC 工作台的数量
E_i	不同 CNC 工作台位置配置的路程花费期望
η	因故障发生降低的系统作业效率比
μ	因工序增加降低的系统作业效率比

4. 任务（1）的求解

4.1 问题分析

智能控制系统的部署有两个部分。首先，当系统开始工作时，每个 CNC 上均没有加工任务，RGV 需为每个 CNC 装配生料，之后，系统正式进入动态调整环节。在指定时间内，智能控制系统的工作流程如下：

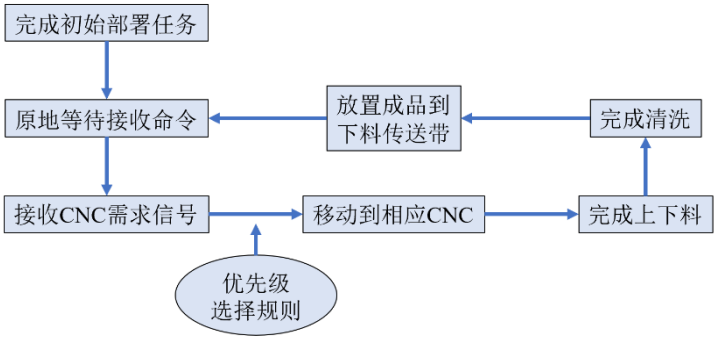


图 1：智能控制系统的工作流程图

针对不同的系统参数，若产品的加工需经过两道工序，则计算机数控机床 CNC 的配置会存在差异。此外，CNC 有 1%的概率出现故障，此时不但在该 CNC 生产线的产品作废，同时该 CNC 还需进入 10 至 20 分钟的等待修复时间。对此，考虑智能控制系统的一般情况，任务（1）的建模流程图如下：

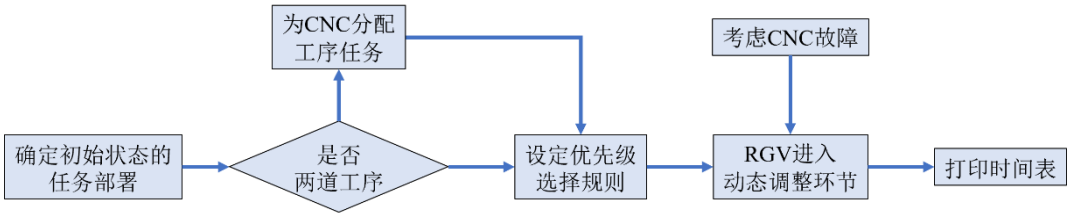


图 2：任务（1）的建模流程图

4.2 模型的准备

4.2.1 初始状态的任务部署

在系统运行的初始阶段，每个 CNC 工作台均处于空闲状态，我们分两种情况进行考虑。第一种情况，若产品的加工只需经过一道工序，则 CNC 在功能上不加以区分，RGV 可按编号顺序为每个 CNC 工作台装配生料，之后便进入动态调整环节。

第二种情况，若产品的加工需要经过两道工序，则首先应为 8 个 CNC 完成工序任务分配。假设我们安排N个 CNC 去完成第一道工序，则有$8-N$个 CNC 负责第二道工序。在初始状态的任务部署工作上，RGV 仅需为负责第一道工序的N个 CNC 工作台装配生料，随后原地等待生料完成第一道加工，再将其运载装配到负责第二道工序的 CNC 工作上。由于两道工序在不同情况下花费的时间不同，因此N的选择会影响智能调控系统的工作效率，N的选择需要经过模型探究。

4.2.2 系统优先级选择规则

当 RGV 完成一项工作后，若只有一个 CNC 发出需求信号，则 RGV 直接前往相应 CNC 所处的位置继续作业即可，但若有多个 CNC 同时向 RGV 发出需求信号，则存在调度系统优先级选择规则的问题。模型的优先选择规则遵循下面原则：

- **原则一：目标距离最短原则**(Distance)

当有编号为$a_1, a_2, \cdots, a_n$的 CNC 工作台同时向 RGV 发出需求信号时，根据目标距离最短原则，RGV 先计算各需求 CNC 工作台到自身位置的距离$d_i(i = 1,2, \cdots n)$，随后选择距离自己最近的 CNC 工作台位置并前往。

- **原则二：奇数优先原则**(Odd)

该原则的优先级低于原则一的优先级。若两台发出需求信号的 CNC 工作台与 RGV 具有相同的距离，则 RGV 再遵循奇数优先原则，先完成编号为奇数的 CNC 工作台的物料装配任务再进行下一轮优先级的判断。这是因为 RGV 为奇数 CNC 工作台完成装配物料的时间更短。

- **原则三：编号最小原则**(Minimum)

该原则的优先级低于原则二的优先级。若两台发出需求信号的 CNC 工作台与 RGV 具有相同的距离，且它们的编号的奇偶性相同，则 RGV 再遵循编号最小原则。先完成编号最小的 CNC 工作台的物料装配任务再进行下一轮优先级的判断。

- **原则四：中心原则**(Central)

该原则的优先级低于原则二的优先级，与原则三同级。若两台发出需求信号的 CNC 工作台与 RGV 具有相同的距离，且它们的编号的奇偶性相同，则 RGV 优先前往靠近智能控制系统中央的 CNC 工作台为其完成装配任务，之后再进行下一轮优先级的判断。该原则能保证 RGV 在之后工作时使其在路径行驶上的时间花费的期望最少。

综上，模型对于优先级的选择规则可以分为两种方案。第一种方案由原则一、原则二和原则三组成，称为 DOM 方案；第二种方案由原则一、原则二和原则四组成，称为 DOC 方案。DOM 和 DOC 方案在处理大部分 CNC 工作台需求信号时的优先级时方法相同，但在两台发出需求信号的 CNC 工作台与 RGV 具有相同的距离，且它们的编号的奇偶性相同时，DOM 和 DOC 执行的优先级不相同。

例如，当 RGV 停在轨道上的第二节位置时（此刻 RGV 为 CNC3 或 CNC4 提供装配服务），在完成当前作业后，RGV 收到 CNC2 和 CNC6 的需求信号，如下图：

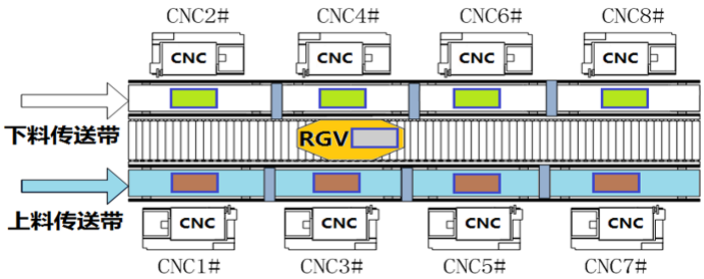


图 3：DOM 和 DOC 优先级选择规则示意图

若系统采用的优先级方案为 DOM，则下一步 RGV 将为 CNC2 提供服务，若优先级方案为 DOC，则下一步 RGV 将会为 CNC6 提供服务。

4.2.3 动态调度模型参数

假设每个 CNC 工作台的工作的状态有四种模式，用$C_n^i(\mathrm{n}=1,\cdots,8,\ i=1,\cdots,4)$表示编号为$n$的 CNC 工作台处于$i$状态，状态数值的定义及其含义如下：

$$\begin{cases} C_n^1=1, & \text{CNC处于空闲状态，等待装配生料} \\ C_n^2=2, & \text{CNC处于空闲状态，需完成上下料} \\ C_n^3=3, & \text{CNC处于工作状态，正在加工} \\ C_n^4=4, & \text{CNC处于故障状态，等待维修} \end{cases} \tag{4.2.1}$$

状态 1 处于任务部署初始状态的阶段，状态 2 至 4 处于智能调度系统的动态调整环节。RGV 收到处于状态 2 的 CNC 的需求信号，便根据优先级选择规则前往完成相应的作业。工作人员收到状态 4 的 CNC 需求信号对其进行检查维修。

对于 RGV，RGV 需要执行四个基本操作 Move，Stay，Load 和 Clean，将它们分别记为 M，S，L 和 C。对于操作 Move，为其分配状态数值及其相应定义如下：

$$\text{Move:}\begin{cases} M_1=1, & \text{RGV移动一个单位} \\ M_2=2, & \text{RGV移动两个单位} \\ M_3=3, & \text{RGV移动三个单位} \end{cases} \tag{4.2.2}$$

对于操作 Load，为其分配状态数值及相应定义如下：

$$\text{Load:}\begin{cases} L_1=0, & \text{RGV进行初始状态的物料装配} \\ L_2=1, & \text{RGV进行第一道工序的物料装配} \\ L_3=2, & \text{RGV进行第二道工序物料的装配} \end{cases} \tag{4.2.3}$$

操作 Stay 表示 RGV 在原地待命，操作 Clean 表示 RGV 进行清洗工作，由于二者操作情况不需分类，因此没有进行多情况赋值。下面分情况对一道工序和两道工序的动态调度模型进行讨论。

此外，对于 RGV 为每个 CNC 工作台进行上下料操作的时间$t_n(n=1,2\cdots,8)$，根据编号的奇偶性，可对t_n定义如下：

$$t_n=\begin{cases} T_0, & n=2k \\ T_1, & n=2k-1 \end{cases} \quad (k=1,2,3,4) \tag{4.2.4}$$

对于具有两道生产工序的系统而言，需要为 8 台 CNC 工作台分配不同的工序任务，用C_n^{mod}表示编号为n的 CNC 工作台的工序状态。C_n^{mod}取值$0-1$，当$C_n^{mod}=0$，表示该 CNC 用于第一道工序的加工，若$C_n^{mod}=1$，则表示该 CNC 用于第二道工序的加工。并用N_1记所有用于第一道工序加工的 CNC 工作台数量，则有：

$$\sum_{n=1}^{8} C_n^{\mathrm{mod}} = 8-N_1 \tag{4.2.5}$$

上式表明所有用于第二道工序加工的 CNC 工作台数量为$8-N_1$，由于两道工序的加工时间不同，因此N_1的选择将会影响系统的生产效率。同时，模型还需考虑两种不同分工的 CNC 工作台的安装位置。

4.3 模型的建立

4.3.1 一道工序的智能调度模型

在 RGV 完成初始状态的部署任务之后，执行 Stay 操作，此时 RGV 处于原地待命阶段，等待 CNC 工作台发出需求信号。智能系统的调度步骤为：

- **步骤一：RGV 首次接收到需求信号C_n^2**

在完成初始状态的部署任务一段时间后，RGV 会收到来自某个编号为n的 CNC 工作台的第一个需求信号C_n^2，RGV 收到信号后度量自身与该 CNC 控制台间的距离d_n，随后选取相同数值执行相应的 Move 操作前往 CNC 工作台处。

- **步骤二：RGV 进行上下料操作并首次清洗熟料**

RGV 到达相应 CNC 工作台处后，调用数值为L_2的 Load 操作取下 CNC 上的熟料，并将生料装配到 CNC 上。完成后 RGV 立即调用 Clean 操作，将该熟料放入清洗槽中进行清洗，由于是首次清洗熟料，因此清洗槽之前未放置任何待清洗物料。

- **步骤三：RGV 再次接收到需求信号C_n^2**

RGV 完成首次清洗任务后，清洗槽中将留下第一块成料，随后 RGV 会收到编号为n的 CNC 工作台的第二个需求信号C_n^2，RGV 收到信号后计算距离调用 Move 操作前往，并调用数值为L_2的 Load 操作取下 CNC 上的熟料，并将生料装配到 CNC 上。

- **步骤四：RGV 再次清洗熟料并将成料放置到下料传送带**

完成步骤三之后，RGV 立即调用 Clean 操作，将机械爪上刚取下的熟料与清洗槽中的成料交换，进行熟料的清洗工作，并将成料放置到下料传送带，完成一次清洗工作，之后 RGV 清洗槽中都将留下一块成料等待与熟料进行交换完成清洗工作。

- **步骤五：RGV 进入待命状态，等待下一轮命令**

完成上述工作后，RGV 执行 Stay 操作，原地待命，或接收到 CNC 工作台的需求信号后，进行优先级判断，选择相应的 CNC 工作台为其执行装配任务。之后在指定时间内，智能调度系统将重复执行步骤三至步骤五。

对于上述智能调度系统的运作流程，有以下约束条件：

$$F_N^1 - F_N^0 \geq t_1 + t_n^l \tag{4.3.1}$$

即对于某个序号为N的物料，它的下料时间与上料时间差不小于所花费的加工时间与上下料时间的和，通常由于 RGV 不会恰好在原地等待，因此 RGV 在接收到 CNC 的需求信号后，还需判断优先级，再花费路程上的时间，最终才能进行换料操作。此外，对于规定 8 小时时间内的生产线上的最后一件物料，需要满足：

$$F_{N_0+1}^1 + t_n^l + t_0 \leq 28800 = 8\times 60\times 60 \tag{4.3.2}$$

上式第一项是第N_0+1件物料的下料完成时间，此时 RGV 清洗槽中装载着第N_0件成品，当 RGV 完成第N_0+1件物料的下料操作，并为 CNC 装配新的物料后，RGV 需执行 Clean 操作，将第N_0+1件熟料放入清洗槽，并将清洗槽中的第N_0件成品放到下料传送带，从而完成第N_0件产品的生产流程，从第N_0+1件物料的下料完成时间算起，还需经过上下料时间t_n^l和清洗时间t_0，第N_0件成品才加工完成，固有上述约束条件。

4.3.2 两道工序的智能调度模型

与一道工序的初始部署任务有些不同，在两道工序的智能调度模型中，初始状态下 RGV 仅能为$C_n^{mod}=0$的 CNC 工作台装配生料，因为其它$C_n^{mod}=1$的 CNC 工作台仅能用于第二道工序的加工。在 RGV 完成初始状态的部署任务之后，执行 Stay 操作，原地待命，等待 CNC 工作台发出需求信号。智能系统的调度步骤为：

- **步骤一：RGV 首次接收来自$C_n^{mod}=0$的需求信号C_n^2**

在完成初始状态的部署任务后，$C_n^{mod}=1$的 CNC 工作台均会发出C_n^1需求信号，但此时生产线上还未生产出半熟料（即完成了一道工序后的物料）。一段时间后，RGV 首次收到来自$C_n^{mod}=0$的需求信号C_n^2。RGV 收到信号后度量自身与该 CNC 控制台间的距离d_n，随后选取相同数值执行相应的 Move 操作前往 CNC 工作台处。

- **步骤二：RGV 进行上下料操作并前往$C_n^{mod}=1$的 CNC 工作台**

RGV 到达相应 CNC 工作台处后，调用数值为L_2的 Load 操作取下 CNC 上的半熟料，并将生料装配到 CNC 上。完成后 RGV 立即对$C_n^{mod}=1$的 CNC 工作台进行优先级判断，选取优先级最高的 CNC 工作台执行 Move 操作前往。

- **步骤三：RGV 首次为$C_n^{mod}=1$的 CNC 工作台装配半熟料**

RGV 到达指定 CNC 工作台后，将机械爪抓取的半熟料装配至 CNC 工作台上，让 CNC 工作台进行第二道工序的加工。随后 RGV 执行 Stay 操作，原地待命。

- **步骤四：RGV 再次收到来自$C_n^{mod}=0$的需求信号C_n^2**

RGV 再次收到来自$C_n^{mod}=0$的需求信号C_n^2时，执行 Move 操作前往，并调用数值为L_2的 Load 操作取下 CNC 上的半熟料，并将生料装配到 CNC 上。之后，RGV 检测是否有来自$C_n^{mod}=1$的 CNC 工作台发出的C_n^1和C_n^2需求信号。若有，则执行步骤五；若否，则 RGV 执行 Stay 操作，原地待命，直到收到来自$C_n^{mod}=1$的需求信号。

- **步骤五：RGV 再次为$C_n^{mod}=1$的 CNC 工作台装配半熟料**

RGV 确认有空闲的 CNC 工作台用于进行第二道工序的加工时，执行 Move 操作前往，并为其装配半熟料。若需求信号为C_n^1，则完成装配任务后执行 Stay 操作，原地待命。若需求信号为C_n^2，则完成装配任务后立即执行步骤六。

- **步骤六：RGV 进行清洗作业**

当 RGV 首次收到来自$C_n^{mod}=1$的需求信号C_n^2时，说明有熟料加工完成，RGV 完成物料装配工作，从 CNC 工作台取下熟料，立即执行 Clean 操作，清洗熟料使其成为成品。若清洗槽中有成品，则将其放置到下料传送带完成一件成品的生产。随后 RGV 执行 Stay 操作，原地待命。智能调度系统将重复执行步骤四至步骤六。

与一道工序的生产流程类似，两道工序也有以下的约束条件：

$$S_N^1 - F_N^0 \geq t_1 + t_2 + t_{n_1}^l + t_{n_2}^l \tag{4.3.3}$$

上式表明，序号为N的物料的第二道工序下料开始时间与第一道工序的上料开始时间的差不小于第一道工序加工用时t_1、第二道工序加工用时t_2、第一道工序上下料用时和第二道工序上下料用时的和。通常由于 RGV 不会恰好在原地等待，因此 RGV 在接收到需求信号后，还需判断优先级，再花费路程上的时间，最终才进行换料操作。

考虑在规定时间内生产的最后一个产品，必须满足：

$$S^1_{N_0+1}+t^l_n+t_0\le 28800=8\times 60\times 60 \tag{4.3.4}$$

上式第一项是第N_0+1件物料第二道工序的下料完成时间，此时 RGV 清洗槽中装载着第N_0件成品，当 RGV 完成第N_0+1件熟料的下料操作，并为 CNC 装配新的物料后，RGV 需执行 Clean 操作，将第N_0+1件熟料放入清洗槽，并将清洗槽中的第N_0件成品放到下料传送带，从而完成第N_0件产品的生产流程，从第N_0+1件物料的下料完成时间算起，还需经过上下料时间t^l_n和清洗时间t_0，固有上述约束条件。

此外，对于两道工序的智能调度模型，由于两道工序所需花费的时间不相同，因此用于不同工序的 CNC 工作的数量需作出调整。从理论上分析，先忽略 RGV 的操作用时，仅考虑 CNC 两道工序的加工用时t_1和t_2，并假设用于第一道工序加工的 CNC 工作台数量为N_1，则参与第二道工序加工的 CNC 工作台数量为$8-N_2$，考虑简单的供给模型，易得下面关于两道工序的供给关系：

$$t_1\times(8-N_1)=t_2\times N_1 \tag{4.3.5}$$

从式（4.3.5）解出N_1的表达式为：

$$N_1=\frac{8\cdot t_1}{t_2+t_1} \tag{4.3.6}$$

当工序的加工时间远大于 RGV 进行上下料、移动和清洗操作的时间时，N_1的实际最优值越接近理论值。此外，除了确定负责第一道和第二道工序的 CNC 工作台数量之外，不同工序 CNC 工作台的配置位置也会影响系统的工作效率。智能调度系统 CNC 工作台的配置模型可用下面的矩阵表示：

$$\begin{bmatrix} C_1^{\text{mod}} & C_3^{\text{mod}} & C_5^{\text{mod}} & C_7^{\text{mod}} \\ C_2^{\text{mod}} & C_4^{\text{mod}} & C_6^{\text{mod}} & C_8^{\text{mod}} \end{bmatrix} \tag{4.3.7}$$

C_n^{mod}的取值仅能为$0-1$，除去全为 0 和全为 1 的两种情况，共有$2^8-2=254$种可能结果，程序遍历 254 种组合的同时，需满足约束条件（4.3.3）和（4.3.4），该规划问题的目的求矩阵（4.3.7）的最优解使得到：

$$\max N_0 \tag{4.3.8}$$

4.3.3 系统故障模型

在前面模型的基础上，考虑 CNC 工作台发生故障的情形。经统计故障发生的概率为 1%，且每次维修的时间为 10 至 20 分钟。在某 CNC 工作台发生故障时，它会发出需求信号C_n^4，随后工作人员赶往进行处理。且此时此工作台上正在加工的物料报废。

因此，在程序的运行中，需要模拟两个随机变量。第一个随机变量用于表示 CNC 工作台是否发生故障，第二个随机变量用于表示发生故障后 CNC 工作台的维护时间。为此我们用下面的模型描述系统故障的发生和解决。

设某 CNC 工作台负责某道工序的加工总用时为t，并设定随机数ξ生成区间为：

$$1 \le \xi \le 100 \cdot t, \quad \xi \in N^* \tag{4.3.9}$$

设ξ服从区间$[1,100 \cdot t]$上的均匀分布，则当ξ落入到区间$[1,t]$时，认为 CNC 工作台将发生故障，由几何概型知识，易得故障发生的概率P即为：

$$P = \frac{t-1+1}{100 \cdot t-1+1} \times 100\% = 1\% \tag{4.3.10}$$

每当某 CNC 工作台开始进行一项加工任务时，为程序添加加工计时器 Counter，计时器负责计算本次加工用时，当下面条件满足时，认为 CNC 发生故障：

$$\text{Counter} = \xi \tag{4.3.11}$$

模拟故障发生的示意图如下所示：

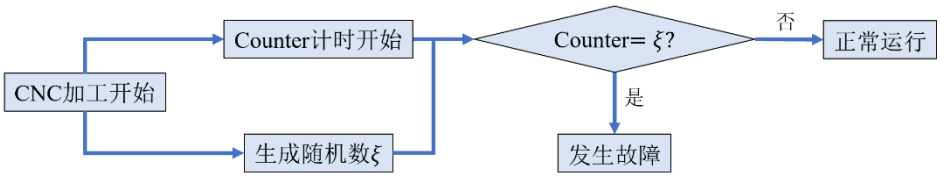


图 4：程序模拟故障发生的示意图

当 CNC 工作台发生故障后，会立即发出C_n^4需求信号，此时工作人员对 CNC 进行维修，维修花费时间为T_n，由于围护时间在 10 至 20 分钟不定，因此设定维修时间T_n是介于区间$[600,1200]$内的随机数，并假设T_n服从均匀分布，即：

$$T_n \sim U[600,1200] \tag{4.3.12}$$

在维修完成后，CNC 将处于空闲状态，此时 CNC 工作台向 RGV 发出C_n^1需求信号，请求添加物料。

4.4 模型的求解

对于只有一道工序的调度系统，仅需分别检验两种不同的优先级规则 DOM 和 DOC 对系统效率的影响，目标函数为：

$$\max N_0 \tag{4.4.1}$$

将取得最优解对应的规则作为一道工序调度系统的优先级规则。而对于具有两道工序的调度系统，模型的求解需解决两个问题：

（1）分配给不同工序的 CNC 工作台数量

（2）负责不同工序 CNC 工作台的最优配置位置

模型将通过理论求解和实验数值求解确定上面两个问题的最优解，以在两道工序的前提下达到式（4.4.1）的要求。不同工序 CNC 工作台数量的配置与工序时间和 RGV 的操作用时有关，而设备的配置位置会影响 RGV 在两道工序间往返调度的时间花费，位置的选择应尽可能的是时间花费的数学期望减小。

4.4.1 DOM 优先级和 DOC 优先级的比较

以第一组数据进行单工序智能调度系统的检验，分别采用 DOM 和 DOC 优先级选择方案，得到实验数据表格如下：

表格 1：第一组不同优先级选择方案的求解结果对比

优先级	DOM 优先级方案			DOC 优先级方案		
序号	CNC 编号	上料时间	下料时间	CNC 编号	上料时间	下料时间
1	1	0	634	1	0	634
2	2	28	687	2	28	687
3	3	79	763	3	79	763
4	4	107	816	4	107	816
5	5	158	892	5	158	892
6	6	186	945	6	186	945
7	7	237	1021	7	237	1021
8	8	265	1074	8	265	1074
…	…	…	…	…	…	…

表格 2：第二组不同优先级选择方案的求解结果对比

优先级	DOM 优先级方案			DOC 优先级方案		
序号	CNC 编号	上料时间	下料时间	CNC 编号	上料时间	下料时间
1	1	0	669	1	0	669
2	2	30	729	2	30	729
3	3	88	817	3	88	817
4	4	118	877	4	118	877
5	5	176	965	5	176	965
6	6	206	1025	6	206	1025
7	7	264	1113	7	264	1113
8	8	294	1173	8	294	1173
…	…	…	…	…	…	…

表格 3：第三组不同优先级选择方案的求解结果对比

优先级	DOM 优先级方案			DOC 优先级方案		
序号	CNC 编号	上料时间	下料时间	CNC 编号	上料时间	下料时间
1	1	0	618	1	0	618
2	2	27	670	2	27	670
3	3	77	745	3	77	745
4	4	104	797	4	104	797
5	5	154	872	5	154	872
6	6	181	924	6	181	924
7	7	231	999	7	231	999
8	8	258	1051	8	258	1051
…	…	…	…	…	…	…

分析求解结果可以得出，对于一般情况下的智能调度系统，DOM 优先级和 DOC 优先级两种方案的求解结果相同，说明在系统的调度过程中，RGV 在优先级的判断上并未出现 DOM 和 DOC 相异层的情况。因此，任何一种优先级方案均可以使得系统得到最优最优解。这两种优先级的选择不会对系统调度效率产生影响。

4.4.2 双工序 CNC 工作台数量配置

式（4.3.5）和（4.3.6）说明了，在理想状况下（即不考虑 RGV 操作用时），两道工序的 CNC 工作台数量的配置应该为：

$$N_1 = \frac{8 \cdot t_1}{t_2 + t_1} \tag{4.4.2}$$

但对于实际情况而言，RGV 的操作时间不可忽略，且 RGV 的操作时间与加工时间的比例越大，它对上述理论值的影响也就越大，只有当加工时间远大于 RGV 操作时间时，上式和真实情况下的最优解才会比较接近。

对于第 1 组、第 2 组和第 3 组数据，理论上第一道工序的 CNC 工作数量N_1为：

$$\begin{cases} \text{第1组：} N_1 = \dfrac{8 \cdot 400}{378 + 400} = 4.11 \approx 4 \\ \text{第2组：} N_1 = \dfrac{8 \cdot 280}{500 + 280} = 2.87 \approx 3 \\ \text{第3组：} N_1 = \dfrac{8 \cdot 455}{182 + 455} = 5.71 \approx 6 \end{cases} \tag{4.4.3}$$

现将 RGV 的操作时间加入考虑，修改关系式（4.3.5），首先考虑 RGV 在移动时的平均路程花费T_w，RGV 每次移动操作可前进 1 至 3 个单位长度，设在一般情况下，移动每个距离的概率相同，则平均路程花费T_w的表达式为：

$$T_w = \frac{T_w^1 + T_w^2 + T_w^3}{3} \tag{4.4.4}$$

其次，考虑 RGV 的清洗作业时间t_0，最后，考虑 RGV 上下料时间，我们记负责第一道工序的 CNC 工作台编号为$a_1, a_2, \cdots, a_{N_1}$，记$T_l$为平均上下料用时，则对于第一道工序而言，平均上下料时间为：

$$T_l = \frac{\sum_{i=1}^{N_1} t_{a_i}}{N_1} \tag{4.4.5}$$

设第二道工序的 CNC 工作台的编号为$b_1, b_2, \cdots, b_{8-N_1}$，则平均上下料用时：

$$T_l = \frac{\sum_{i=1}^{8-N_1} t_{b_i}}{8 - N_1} \tag{4.4.6}$$

引入上述 RGV 操作时间，修正式（4.3.5）得到分配给第一道工序的 CNC 工作台的数量N_1应满足：

$$\left(t_1+\frac{\sum_{i=1}^{N_1}t_{a_i}}{N_1}+T_w+t_0\right)\times(8-N_1)=\left(t_2+\frac{\sum_{i=1}^{8-N_1}t_{b_i}}{8-N_1}+T_w+t_0\right)\times N_1 \tag{4.4.7}$$

用修正后的式（4.4.7）再次求解第一道工序的 CNC 共工作台的数量N_1得到：

$$\begin{cases}\text{第1组：}N_1\approx 4\\ \text{第2组：}N_1\approx 4\\ \text{第3组：}N_1\approx 5\end{cases} \tag{4.4.8}$$

式（4.4.8）即为不同工序的 CNC 工作台配置数量的理论解。

4.4.3 双工序 CNC 工作台的最优配置位置

在确定负责不同工序的 CNC 工作台的最优配置数量后，还需确定不同分工的工作台的配置位置，由（4.4.8）的结果，下面讨论第一道工序与第二道工序的数量比为 4:4 的位置配置情况。

考虑三种极端情况，让负责第一道的工序的 CNC 工作台分别分配至系统两侧、系统中间，及第一道工序与第二道工序对称分布，如下图：

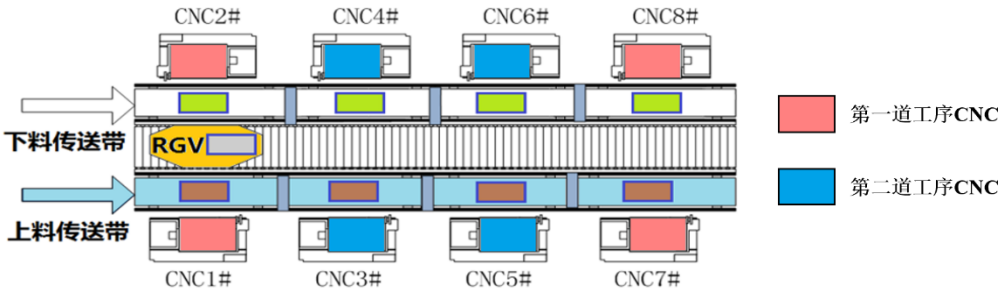


图 5：第一道工序 CNC 分配至系统两侧示意图

考虑理想情况，假设 RGV 顺序为四对 CNC 工作台分别完成物料装配任务，并假设 RGV 初始状况下等可能的出现在导轨的四段不同位置。仅考虑路程花费的期望，不考虑加工时间和上下料时间，在第一种情况下，RGV 工作中路程花费的期望为：

$$E_1=\frac{2\times\left[(6T_w^1+T_w^2)+(7T_w^1+T_w^2)\right]}{4} \tag{4.4.9}$$

上式中$(6T_w^1+T_w^2)$一项表示 RGV 初始位置在第一格时完成任务的时间花费，$(7T_w^1+T_w^2)$一项表示 RGV 初始位置在第二格时完成任务的时间花费，由工作台位置分布的对称性，当 RGV 在第三和第四格时，其值与前面讨论的情况相同，故在分子上乘以了 2 倍，最后再乘以每种情况出现的概率四分之一即为所求期望。

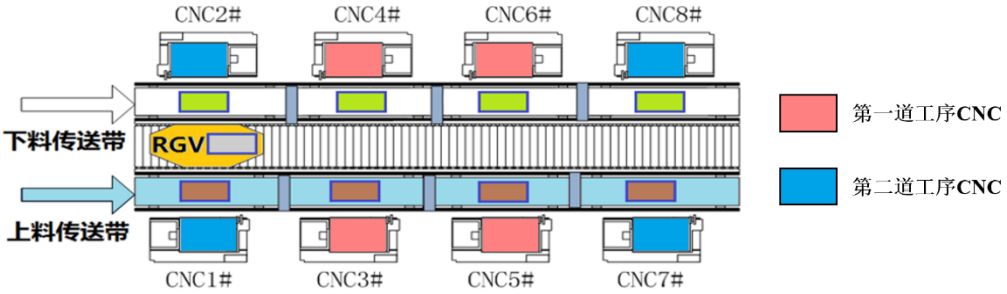


图 7：第一道工序 CNC 分配至系统两侧示意图

同理，求解得到第二种情况的路程花费期望E_2为：

$$E_2 = \frac{2\times\left[(7T_w^1 + T_w^2) + (6T_w^1 + T_w^2)\right]}{4} \tag{4.4.10}$$

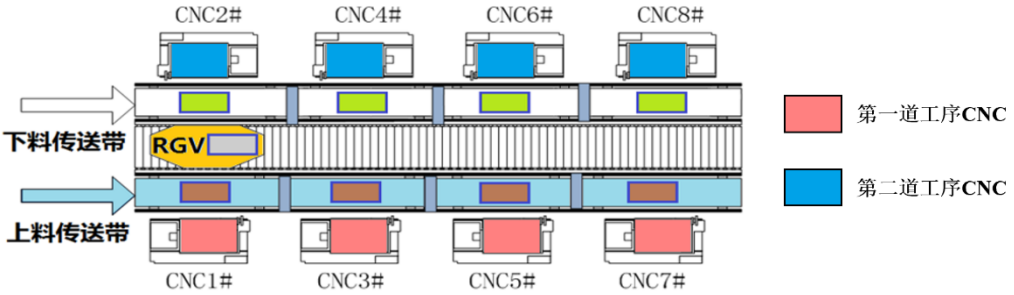


图 6：第一道工序与第二道工序 CNC 对称分布示意图

求解得到第三种情况的路程花费期望E_3为：

$$E_3 = \frac{2\times\left[(3T_w^1) + (2T_w^1 + T_w^2)\right]}{4} \tag{4.4.10}$$

对比式（4.4.8）至（4.4.10），能够得到：

$$E_3 < E_1 = E_2 \tag{4.4.11}$$

上式说明在一般情况下，应该让两种不同工序的对称放置，这样 RGV 在从第一道工序的 CNC 工作台上取下半熟料后，可以直接原地等候将半熟料直接装配到对面负责第二道工序的 CNC 工作台上，最大程度的减少了 RGV 在路程上的开销。

若再加入加工时间，上下料时间等 RGV 的操作用时，则不同的加工工序分配给奇偶性不同的 CNC 工作台也会对系统的运行效率产生影响。

4.4.2 节和 4.4.3 节都是从分析的角度解决矩阵（4.3.7）的最优解问题，解决问题的目的都是尽量减少 RGV 在等待时间和路程时间上的花费，从而提高系统效率，在规定时间内尽可能的生产更多成品。程序还将通过三组数据遍历$0-1$矩阵的所有可行解，并以（4.3.8）为目标函数寻找矩阵的最优解，以数值方法求解模型，再和分析解的结果进行对比。

5. 任务（2）的求解

5.1 问题分析

通过任务（1）中建立的调度模型，可以利用 3 组数据求解得到 RGV 的调度方案，同时能够比较不同优先级选择方案对系统工作效率的影响。同时，还需确定对于双工序情况下的不同工序 CNC 工作台的最优配置数量和最优配置位置。我们以系统平均每小时完成的生产件数作为系统的作业效率。任务（2）的建模流程图如下：

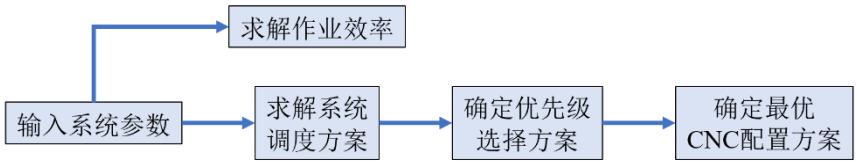


图 8：任务（2）的建模流程图

5.2 模型的求解

首先定义每一组系统的作业效率η为平均每小时完成的成品数量：

$$\eta = \frac{N_0}{8} \tag{5.2.1}$$

模型求解得到各种情况下的系统作业效率为：

表格 4：各种情况和不同系统参数下的系统作业效率

情况分类	第 1 组	第 2 组	第 3 组
单工序无故障	44.625	42.125	45.875
双工序无故障	29.5	25.25	30.125
单工序有故障	43.625	40.75	44.75
双工序有故障	28.875	24.125	29.75

通过对所有情况的求解，发现使用两种优先级选择方案 DOM 和 DOC 所得到的结果完全相同。分析两种优先级规则示意图如下：

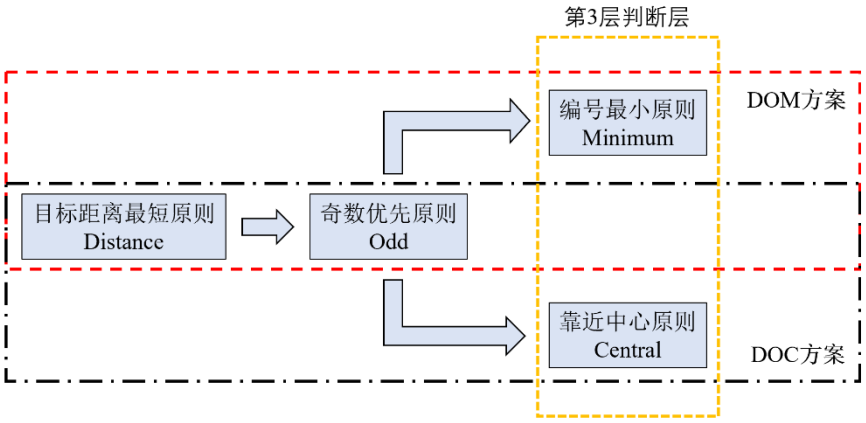


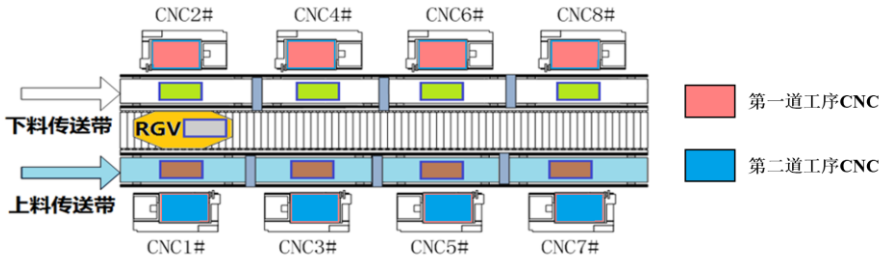
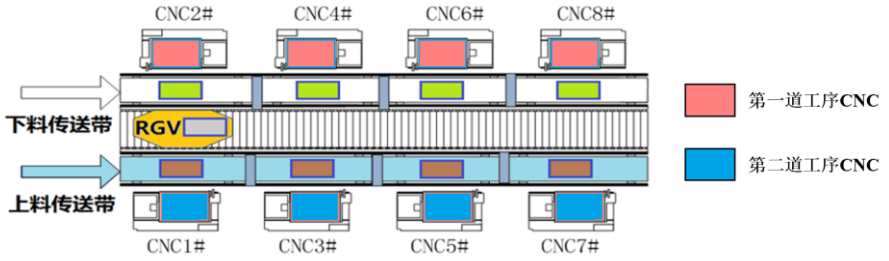
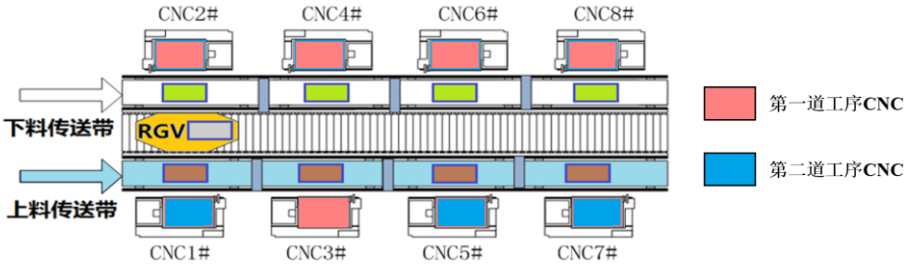
图 9：DOM 和 DOC 优先级方案规则示意图

两种不同优先级方案仅在第 3 层判断层有区别，而最终的求解结果显示两种优先级方案并无差别。说明有以下两种可能，

- 在智能调度系统的运行过程中，RGV 接收到需求信号的所有情况并不会进入到第 3 层判断条件，在前两层已经可以做出优先级判断
- 在 RGV 进行第 3 层优先级判断的时候，DOM 和 DOC 两种方案的原则得到同一解

此外，模型求解得到的双工序 CNC 工作台最优配置方案如下表：

表格 5：不同系统参数下双工序 CNC 工作台最优配置

系统参数	双工序 CNC 工作台最优配置方案
第 1 组	
第 2 组	
第 3 组	

上述求解结果对比 4.4.2 节和 4.4.3 节的理论分析解，分配给不同工序的 CNC 工作台的数量符合理论分析解。即我们得到最优的 CNC 工作台数量配置为：

$$\begin{cases} \text{第1组：} N_1 = 4 \\ \text{第2组：} N_1 = 4 \\ \text{第3组：} N_1 = 5 \end{cases} \tag{5.2.2}$$

且 CNC 工作台的配置位置按照式（4.4.11），需要做到两道不同的工序对称分布才能够使得 RGV 路程花费的期望最小，因此 3 组不同系统参数下，负责第一道工序和第二道工序的 CNC 工作台均一对一对称分布。

5.3 求解结果分析

系统对故障发生的概率及发生故障的 CNC 工作台的维护时间是随机的，因此每次模拟得到的系统的作业效率会不相同，但我们可以计算系统发生故障时降低的工作效率比，记φ为系统因故障发生时降低的工作效率比，用η_0表示无故障发生情况下的系统作业效率，η_1表示有故障发生情况下的系统作业效率，定义φ为：

$$\varphi = \frac{\eta_0 - \eta_1}{\eta_0} \times 100\% \tag{5.3.1}$$

由表格 4 中的作业效率，计算单、双工序的降低的工作效率比为：

表格 6：因故障发生降低的工作效率比

类别	第 1 组	第 2 组	第 3 组
单工序	2.241%	3.264%	2.452%
双工序	2.119%	4.455%	1.245%

此外，考虑在相同系统参数下，由单工序变为双工序，系统降低的工作效率μ，用η_0表示单工序系统作业效率，η_1表示双工序系统作业效率，定义μ为：

$$\mu = \frac{\eta_0 - \eta_1}{\eta_0} \times 100\% \tag{5.3.2}$$

由表格 4 中的作业效率，计算因工序变化降低的工作效率比为：

表格 7：因工序增加降低的工作效率比

类别	第 1 组	第 2 组	第 3 组
无故障	33.894%	40.059%	34.332%
有故障	33.811%	40.798%	33.520%

分析表格 4、表格 6 和表格 7 的结果我们得到：

- 工序从单工序变为双工序时，3 组系统参数下的系统作业效率都降低 33%以上，其中第 2 组的作业效率降低最为显著，降低 40%左右
- 当系统的作业情况从无故障转变为有故障时，3 组系统参数下的作业效率都有降低，第 1 组和第 3 组降低的效率约为 1%至 2%，而第 2 组参数下因故障降低的效率在 4%左右

分析上面结果产生的原因，从题目的表 1 中发现，第 2 组系统参数下，RGV 在进行移动，上下料，清洗所花费的时间均超过第 1 组和第 3 组数据，因此在第 2 组参数的条件下，智能调度系统在路程，上下料和清洗上的时间开销更大。当工序从单工序变为双工序后，致使系统的作业效率下降达到 40%之多。综上，为了提高该智能调度系统的作业效率，尽可能提高 RGV 的操作速度，减少加工外的时间开销是关键之一。

6. 模型分析

6.1 模型的优缺点

模型的缺点：

在建立智能调度系统模型时，我们做出了“RGV 在未收到 CNC 需求信号时，无法移动，只能原地待命”的假设。该假设意味着 RGV 无法记录各 CNC 工作台的加工耗时，每当 RGV 完成一项任务时，若未收到需求信号，则原地不动。而由于 CNC 的加工用时远大于 RGV 进行移动、上下料和清洗的操作用时，因此在系统的运行时间内，RGV 很大概率上会处于空闲静置状态。

而去除该假设，若假设 RGV 能够为每台 CNC 的加工耗时进行计时，那么 RGV 在完成当前任务后，能计算各 CNC 加工完成度，并提前前往下一个即将完成的 CNC 工作台位置处等待其需求信号指令。对比无法移动的模型，该模型显得更加智能，减少了 RGV 的空闲等待时间，也能较大地提高系统作业效率。

模型的优点：

模型在智能调度系统的安排上考虑了两种不同的优先级选择方案 DOM 和 DOC，通过求解发现在两种不同的优先级条件下，系统的工作效率完全相同。此外，模型考虑了双工序情况下，负责不同工序的 CNC 工作台的最优配置数量和配置位置，分别从分析解和数值解的角度相互印证，得到最优配置方案。

6.2 模型误差分析

分析得出，模型的主要误差来源于以下几个方面：

- 未考虑生料在上料传送带上传送用时。而事实上，当 CNC 完成生料加工发出需求信号时，生料才会被放上传送带传送到相应位置，RGV 也才能进行上下料操作
- 因为未考虑 RGV 对 CNC 加工耗时的记录功能，且 RGV 在完成当前任务后若没有收到 CNC 需求信号只能原地待命。若 RGV 具有计时功能，则它能够提前前往即将完成的 CNC 工作台处等待需求信号。因此该模型的最优解与现实有差距。

6.3 模型的推广

本文基于 DOM 与 DOC 两种优先级比较规则对 RGV 智能加工系统制定了高效的调度方案，其中多原则对比的动态规划思想可运用到更多的调度问题中去；对于问题本身，该模型在 CNC 数量不太大、加工工序数量不太多的情况下在制定调度方案时有较为良好的表现，但如果 CNC 的数量增多到一定程度时，在确定最优工序配置的环节效率会相应降低，这时就需要增加 RGV 的数量建立更加高效的调度系统。

7. 参考文献

[1] 任晓军.AGV 在物流工程中的应用研究[J].科技经济导引, 2016 (31).
[2] 司守奎, 孙玺菁. 数学建模算法与应用[M]. 国防工业出版社, 2011.
[3] Dossey J. A First Course in Mathematical Modeling (P) by Frank R. Giordano; Maurice D. Weir[J]. Mathematics Teacher, 1985.
[4] 佚名. 基于动态优先级算法的智能 RGV 动态调度策略[J]. 通讯世界, 2019(4).

附录

由于代码较为相似，这里给出第 1 组参数下的 C++代码如下：

```
#include<iostream>
#include<cstdio>
#include<string>
#include<cstring>
#include<vector>
#include<algorithm>
#include<fstream>
#include<map>
using namespace std;
#define WAIT0     0          // 没有熟料的等待
#define WAIT1     1          // cnc 上有一块熟料
#define PROC      2
#define REPAIR    3

int mes;              // 等待 rgv 处理的 cnc 的个数，也即当前 rgv 收到的消息数
struct CNC{
    int id;         // 编号{1,2,3,4,5,6,7,8}
    int pos;        // 位置{1,2,3,4}
    int state;    // 状态{0 或 1：等待 cnc, 2：加工中, 3：修理中}，其中 0 表示 cnc 上没有板
    int time;     // 如果处于 0 或 1 状态，则表示已经等待了的时间，如果处于 2，表示距离加工完成所需的时间
    int num;        // 加工物料的编号
    int proc_time = 560;   // 加工物料所需时间

    // 构造函数
    CNC(int i, int pt) : id(i), pos((i+1)/2), state(WAIT0), time(0), num(0), proc_time(pt)    {}
    CNC() : CNC(0, 560) {}          // 委托构造函数
```

```
    // 模拟 cnc 处理 t 秒（加工或者等待或者修理）
    // 输入：t   表示处理的时间
    // 返回：int  处理完成后的状态
    int proc(int t)
    {
        if(state < 2)
            time += t;
        else if(state == PROC){
            time -= t;
            if(time <= 0){
                state = WAIT1;
                time = -time;
                mes++;
            }
        }
        return state;
    }

    // 模拟状态改变，即更换物料
    // 输入：n        新放上的物料的编号
    // 返回：int     处理完成后的状态
    int change(int n)
    {
        num = n;
        state = PROC;
        time = proc_time;
        mes--;
        return state;
    }
};

struct RGV{
    int time;                  // 从开始运行到当前的时间
    int pos;                   // 位置{1,2,3,4}
    int n1, n2;               // n2 清洗槽中存放的物料编号 {0,1,2,...}，其中 0 表示没有物
料, n1 生料
    int move_time[4] = {0, 20, 33, 46};
    int change_time[10] = {0, 28, 31, 28, 31, 28, 31, 28, 31, 0};
    const int clean_time = 25;

    // 构造函数
    RGV(int *a, int *b, int cl) : time(0), pos(1), n1(0), n2(0), clean_time(cl)
    {
        for(int i=0;i<4;i++)
```

```
            move_time[i] = a[i];
        for(int i=0;i<10;i++)
            change_time[i] = b[i];
    }
    RGV() : time(0), pos(1), n1(0), n2(0) {}

    // change 模拟 rgv 从当前位置去到 n 号 cnc 机器并完成上下料这一系列工作
    // 输入：n   表示待处理的 cnc 编号
    // 返回：int   这一过程的时间
    int change(int n)
    {
        n1++;                   //拿来一块新的生料
        int total_time = change_time[n];
        time += total_time;
        return total_time;
    }

    // move_to 模拟 rgv 移动
    // 输入：n   表示待处理的 cnc 编号
    // 返回：int 这一过程的时间
    int move_to(int n)
    {
        int t = move_time[abs((n+1)/2 - pos)];
        time += t;
        pos = abs((n+1)/2);
        return t;
    }


    // clean 模拟 rgv 清洗工作
    // 输入:num 此时机械臂上的物料编号
    // 返回：清洗完放上下料传送带的物料的编号
    int clean(int num)
    {
        int finish = n2;
        n2 = num;
        time += clean_time;
        return finish;
    }
};

// 物料
struct matter{
    int cnc_number = -1;
```

```
    int start_time = -1;
    int leave_cnc = -1;
    int finish_time = -1;
} mt[1000];

int find_cnc(vector<CNC>, int p);

int problem1(RGV &rgv, vector<CNC> &cnc, const string file_name)
{
    memset(mt, -1, sizeof(mt));              // 这里-1 是对的，但-2 或者其他可能不对了，因为单位是 char
    ofstream out(file_name);
    mes = 8;
    while(rgv.time <= 28800){
        if(mes){                                  // 有 cnc 处于等待状态
            int n = find_cnc(cnc, rgv.pos);
            int t = rgv.move_to(n);
            for(int i=1;i<=8;i++)
                cnc[i].proc(t);
            if(!(n%2) && cnc[n-1].state < 2)
                n--;            // 如果到了这里发现，这个偶数号机器对面的机器也好了，那就先处理奇数号的
            mt[rgv.n1 + 1].start_time = rgv.time;    // 这里 rgv.n1+1 表示接下来即将放上去的新的物料的编号
            mt[rgv.n1 + 1].cnc_number = n;
            int down = cnc[n].num;          // 取下的熟料
            mt[down].leave_cnc = rgv.time;
            t = rgv.change(n);
            for(int i=1;i<=8;i++)
                cnc[i].proc(t);
            cnc[n].change(rgv.n1);
            if(down){                             // 如果 down 不为 0，说明取下了熟料，要清洗
                int finish = rgv.clean(down);
                for(int i=1;i<=8;i++)
                    cnc[i].proc(rgv.clean_time);
                mt[finish].finish_time = rgv.time;
                if(finish){                   // 如果 finish 不为 0，说明从清洗槽中取出了成料
                    printf("%d finish at %d s\n", finish, rgv.time);
                    cout << finish << " " << rgv.time << endl;
                }
            }
        }
```

```
            else{                                   //所有 cnc 都在加工，没有等待的 cnc
                int min_t = 600;                    //下面先找到最快哪个 cnc 能加工好，因为
最大不可能超过 600，所以先设置为 600
                for(int i=1;i<=8;i++)
                    if(cnc[i].time < min_t)
                        min_t = cnc[i].time;
                for(int i=1;i<=8;i++)
                    cnc[i].proc(min_t);
                rgv.time += min_t;
            }
        }
        for(int i=0;i<1000;i++){
            if(mt[i].cnc_number > 0){
                out << mt[i].cnc_number << " " << mt[i].start_time << " " << mt[i].leave_cnc
<< " " << mt[i].finish_time << endl;
            }
        }
        out.close();
        return 0;
}

bool cmp(const CNC &a, const CNC &b, int p)
{
        if(a.state > 1) return false;
        else if(b.state > 1) return true;
        else{
            if(abs(a.pos - p) < abs(b.pos - p)) return true;
            else if(abs(a.pos - p) > abs(b.pos - p)) return false;
            else{
                if((a.pos % 2) && !(b.pos % 2)) return true;
                else if(!(a.pos % 2) && (b.pos % 2)) return false;
                else{
                    return (a.time > b.time);
                }
            }
        }
}

int find_cnc(vector<CNC> t, int p)
{
        auto it = min_element(t.begin() + 1, t.end() - 1,
                              [p](const CNC &a, const CNC &b){return cmp(a, b, p);});
        return (it - t.begin());
}
```

```
int main()
{
    int move_time[3][4] = {{0, 20, 33, 46}, {0, 23, 41, 59}, {0, 18, 32, 46}};
    int change_time[3][10] = {{0, 28, 31, 28, 31, 28, 31, 28, 31, 0},
                              {0, 30, 35, 30, 35, 30, 35, 30, 35, 0},
                              {0, 27, 32, 27, 32, 27, 32, 27, 32, 0}};
    int clean_time[3] = {25, 30, 25};

    RGV rgv1(move_time[0], change_time[0], clean_time[0]);
    vector<CNC> cnc(10);
    for(int i=1;i<=8;i++){
        CNC t(i, 560);
        cnc[i] = t;
    }
    problem1(rgv1, cnc, "output1-1.txt");          // 解决第一问第一组

    RGV rgv2(move_time[1], change_time[1], clean_time[1]);
    for(int i=1;i<=8;i++){
        CNC t(i, 580);
        cnc[i] = t;
    }
    problem1(rgv2, cnc, "output1-2.txt");          // 解决第一问第二组

    RGV rgv3(move_time[2], change_time[2], clean_time[2]);
    for(int i=1;i<=8;i++){
        CNC t(i, 545);
        cnc[i] = t;
    }
    problem1(rgv3, cnc, "output1-3.txt");          // 解决第一问第三组
    return 0;
}
```