基于动态规划与单亲遗传算法的智能调度模型

摘要

作业车间调度优化问题是典型的 NP-hard 问题，高效的调度算法对降低生产成本具有重要意义。本文针对智能加工系统多种情况下的调度问题，基于决策论与动态规划的思想，以合理利用机床资源、提高成料产量为目标，建立了动态最短路径模型，并结合改进后的单亲遗传算法对问题进行求解。

针对任务一：对于仅一道工序的物料加工情况进行分析，将全局最短路问题转化为对局部路径的预判与规划，选取当前时刻最快完成任务的 k 个机床，基于决策论的思想建立了动态规划模型。考虑未来一段时间内所有可能的路径，以最短路径的起始位置作为 RGV 引导车调度的目标位置并重复过程即可实现生产效率的最大化。对于两道工序的物料加工情况，经过分析发现生产效率与负责两项工序的机床数量、位置紧密相关，运用改进后的单亲遗传算法计算出最优的机床位置分配，在此基础上进一步扩充动态路径规划模型。上述建立的动态最短路径模型对于可能发生故障的情形具有很强的鲁棒性。

针对任务二：将题目所给的三组数据分别代入上述模型，运用 python 进行编程求解。得到一道工序加工情况下的三组数据的物料最大加工数分别为：382，359，392；两道工序加工情况下结合单亲遗传算法，给出了机床的最优分布及物料加工数：

- 第一组的最优机床分布为 [1,2,2,1,2,1,2,1]，加工物料数为 252 个
- 第二组的最优机床分布为 [1,2,2,2,1,2,2,2]，加工物料数为 197 个
- 第三组的最优机床分布为 [1,1,2,1,2,1,1,2]，加工物料数为 239 个

考虑故障情况，使用随机数进行仿真模拟，其中一种情况为一道工序加工情况下的三组数据的物料最大加工数分别为：366，354，381；两道工序加工情况下的三组数据的物料最大加工数分别为：250，197，242。基于模型的求解结果，分析出每种情况下的 RGV 引导车的调度策略。为衡量系统的工作效率及算法有效性，我们选取了一个班次内的机床平均空闲时间作为指标，同时，对故障情形进行多次独立重复实验，得到结论：模型的鲁棒性高，实用性很强。

最后，对本文模型进行了合理的检验，检验结果显示，本文提出的动态最短路径规划模型具有生产效率高、鲁棒性强等特点，可以应用于实际的生产生活中。

关键词：车间调度　最短路径规划模型　单亲遗传算法　动态规划　仿真测试

目录

一、问题重述

1.1 问题背景

在市场竞争日益激烈的今天，生产制造企业面临着诸多挑战，如同类产品数量众多，种类丰富，价格低廉等，那么，如何在保证产品质量的前提下，降低生产成本、缩短生产周期以提升企业自身的竞争力，逐渐成为工业制造领域中一项重要的课题。

在工业制造领域中，调度占据着重要地位，调度系统的效率会直接影响到企业生产效率和生产成本。传统的工业生产过程中，通常采用人工作业的方式进行调度，即人工将产品原料由传送带放置于各工作台并在产品加工完成后将产品取出回收，但人工作业存在着诸多问题，如管理及工资成本高、工人疲劳会降低工作效率及任务完成的精度等，因此现代工业生产中大部分企业都引进了自动化生产，运用智能系统进行调度，大大降低了生产成本且工作效率得到了显著提高。

作业车间调度优化问题就是工业自动化过程中研究自动化调度中衍生出来的重要问题，运用算法对生产任务进行合理的规划，尽可能运用更少的时间完成更多的任务，以提高生产效率达到利益的最大化。但作业车间调度优化问题是典型的 NP-hard 问题，随着问题影响因素越多，其求解过程会变得更加复杂，最优解的计算量也会变得十分庞大，且生产中存在着诸多干扰因素如机器故障等，因此，构建一个高效合理的模型求解此问题会对工业制造领域大有裨益。

1.2 问题提出

工业生产领域中，企业一般会使用智能加工系统对工作任务进行调度以提高生产效率。一种智能加工系统由 8 台计算机数控机床（Computer Number Controller，CNC 机床）、1 辆轨道式自动引导车（Rail Guide Vehicle，RGV 引导车）、1 条 RGV 引导车直线轨道、1 条上料传送带、1 条下料传送带等附属设备组成。RGV 引导车是一种无人驾驶、能在固定轨道上自由运行的智能车，可以根据指令自动控制移动方向和距离，并自带一个机械手臂、两只机械手爪和物料清洗槽，能够完成上下料及清洗物料等作业任务。

在实际的工业生产过程中一般存在下面的三种具体情况：

(1) 一道工序的物料加工作业情况，每台 CNC 机床安装同样的刀具，物料可以在任一台 CNC 机床上加工完成；

(2) 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 机床依次加工完成；

(3) CNC 机床在加工过程中可能发生故障（据统计：故障的发生概率约为 1%）的情况，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，故障排除后即刻加入作业序列。要求分别考虑一道工序和两道工序的物料加工作业情况。

基于以上三种情况，需要解决下列两个问题：

(1) 对一般问题进行研究，给出 RGV 引导车动态调度模型和相应的求解算法；
(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 引导车的调度策略和系统的作业效率，并将具体的结果填入附件 2 的 EXCEL 表中。

二、模型假设

(1) 传送带运行时间极小，模型中不考虑传送带运行时间；
(2) 在整个工作过程中，RGV 引导车不会发生故障；
(3) RGV 引导车只会对已加工好的熟料进行清洗；
(4) RGV 引导车在已接受到的需求指令中完成决策后会给对应的 CNC 机床发送信号，在移动到目标的过程中不会中途改变移动目标；
(5) 每台 CNC 机床故障发生相互独立，故障率均为 1%；
(6) 两道工序的物料加工作业情况中，每台 CNC 机床在一个班次内不更换道具，即全程只能参与一道工序的加工。

三、符号说明

符号	定义
t	当前班次已运行时间
n_i	CNC_i 机床当前加工完成的物料个数
n_{tol}	该班次中当前加工完成的物料总数
$w_{i,t}$	CNC_i 机床 t 时刻当前执行任务的剩余时间
f_0	CNC 机床加工完成一个一道工序的物料所需时间
f_1	CNC 机床加工完成一个两道工序物料的第一道工序所需时间
f_2	CNC 机床加工完成一个两道工序物料的第二道工序所需时间
I_t	t 时刻 RGV 引导车所处位置
M_i	RGV 引导车移动 i 个单位所用时间
T_i	RGV 引导车为 CNC_i 机床一次上下料所需时间
T_w	熟料清洗及运出时间
CNC_i	编号为 i 的CNC 机床
F_r	机床剩余维修时间

注：文中出现次数较少的参数在使用时注明

四、智能调度系统的模型建立

4.1 问题分析

题目以智能加工系统为背景，介绍了智能加工系统的组成及作业流程，问题一要求我们基于实际生产中的三种基本情况，对一般情况进行研究，给出 RGV 引导车动态调度模型及相应的求解算法。

基于附件一的 RGV 引导车工作流程，我们首先考虑一般情况下不同情形的 RGV 引导车的调度情况。附件一指出，各个传送带可以连动，也可以进行独立运动，因此可以认为 CNC 机床接收到 RGV 引导车的反馈信号后，相应的传送带会立即传送出未加工的物料，对上料过程不会产生影响，而在下料完成后，传送带也会立即将熟料送出，不会影响下一次的下料过程。于是，在模型建立的过程中，认为传送带对整个加工过程不产生任何影响。

在进一步分析的过程中发现，实际调度中会遇到接收到多个需求信号的情形，因此我们需要解决多目标的决策问题，同时，传统的智能加工系统中由于 RGV 引导车会存在无信号时停在原地进行等待的情况，会造成一定时间的损失，因此，可以在系统中加入预判过程，提前移动至下个完成目标提高工作效率。具体的调度分析过程如下：

(1) 情形一：未充分利用车床资源

首先我们考虑一种特殊情形，即当 RGV 引导车在给第 i 台机床上料结束之前，已上料的 CNC 机床已完成工作并给其发出需求信号，则为达到工作效率即生产物料数的最大化，会出现 RGV 引导车返回至前面 CNC 机床的情形，即 RGV 引导车会在邻近 CNC 机床中不断循环调度，后面机床则会一直处于空闲状态。为便于读者理解，首先对此情形进行定性分析，设定特定的情形，绘制出不同时刻系统工作状态的图像进行 RGV 引导车调度过程的演示。

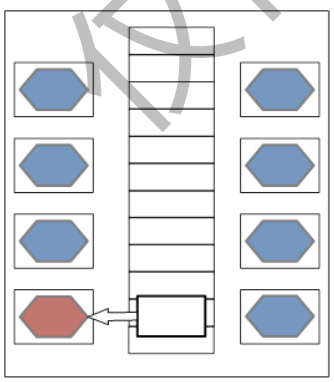


图 4-1 初始时刻

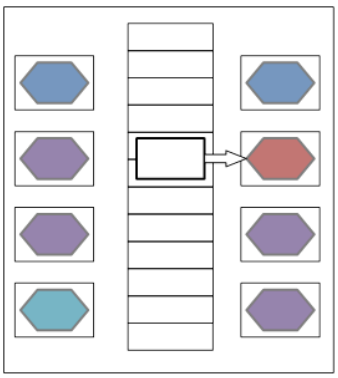


图 4-2 接收到需求信号

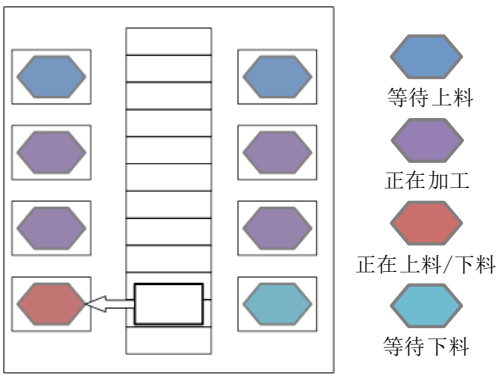


图 4-3 返回初始位置

如图 4-1 所示，RGV 引导车首先对 CNC_1 机床进行上料，其余机床处于等待上料即空闲状态；

如图 4-2 所示，RGV 引导车按编号依次对 CNC 机床进行上料，在给 CNC_6 机床进行上料时，CNC_1 机床完成加工，同时向 RGV 引导车发出需求信号；

如图 4-3 所示，由于系统需要尽可能多的完成物料加工的数量，因此在完成 CNC_6 机床上料任务后，RGV 引导车会移动至 CNC_1 机床进行上下料工作，而不会移动至 CNC_7 机床处进行上料工作，并且在为 CNC_1 机床进行上料工作的同时，CNC_2 机床完成加工。

随后，RGV 会重复上述步骤，在 CNC_1 机床与 CNC_6 机床间进行调度，CNC_7 机床与 CNC_8 机床则会一直处于空闲状态。

接下来我们研究此种情形中各个变量需要满足的关系式，对此情形进行定量描述与分析。基于以上的分析得到，当 CNC_1 机床完成物料加工，向 RGV 引导车发送需求信号时，RGV 引导车完成当前任务则会立即返回，系统将会不断进行循环调度，后面的机床会一直处于空闲状态。因此在 CNC_1 机床物料加工完成前，RGV 引导车还未调度到 CNC_7 机床与 CNC_8 机床之间，计算出各变量所需要满足的关系式：

$$T_1 \cdot (i-1) + T_2 \cdot i + 2i \cdot T_w \geq f_0 \tag{1}$$

注 其中 i 为 RGV 引导车所在的层数，$i = 1,2,3$，T_i 为 RGV 引导车为 CNC_i 机床上下料所需的时间，$i = 1,2$，T_w 为物料清洗时间，f_0 为一道工序物料加工所需的时间。

在此种情形中，为提高系统工作效率，实际应该考虑增加 RGV 引导车进行工作。而在本题背景下，我们只需考虑一台 RGV 引导车的情况，因此在全文中对此种情形不予考虑。

(2) 情形二：同时收到需求信号

更一般的，智能加工系统通电启动后，RGV 引导车开始按编号依次给 CNC 机床进行上料，某种时刻可能会出现在 RGV 引导车给某个 CNC 机床进行上下料一系列操作结束前，接收到多个 CNC 机床发出的需求信号，于是，在 RGV 引导车结束工作时，就涉及到选取哪个 CNC 机床作为下一个工作目标的问题，即工作目标的决策问题。

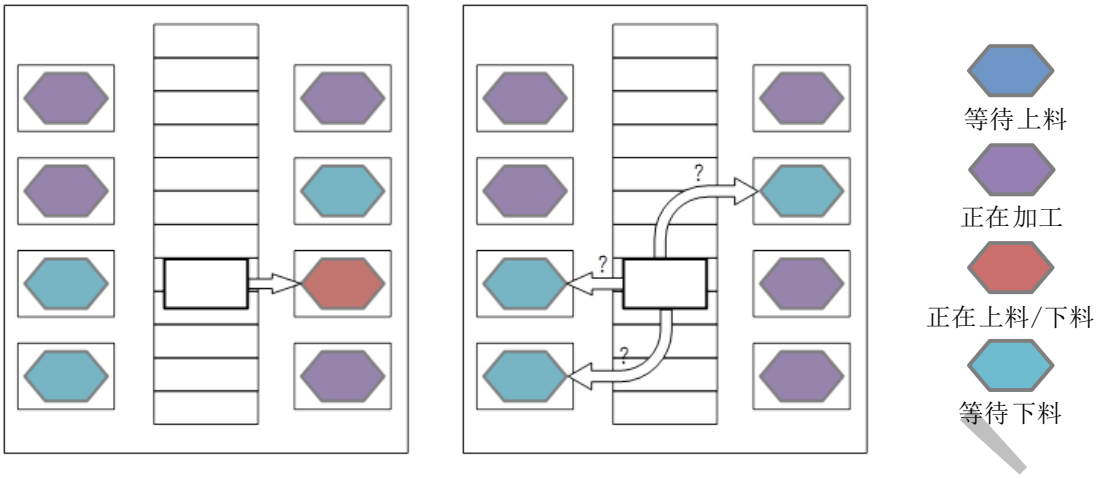


图 4-4 RGV 引导车工作　　**图 4-5 同时收到信号，进行路径决策**

(3) 情形三：RGV 引导车对路径进行预判

由附件一指出，在工作正常情况下，如果某 CNC 机床处于空闲状态，则向 RGV 引导车发出上料需求信号；否则，CNC 机床处于加工作业状态，在加工作业完成时即刻向 RGV 引导车发出需求信号。因此，当没有任何 CNC 机床向 RGV 引导车发出需求信号时，RGV 引导车会在原地等待指令再移动至目标处，在此过程中，此移动时间实际为生产过程中的浪费时间，若 RGV 引导车可以进行一定“预判”，即在无需求信号时预测到下一个任务目标，先行移动到相应的 CNC 机床处，则可以节省一定的时间，提高生产效率(如图 4-6)。

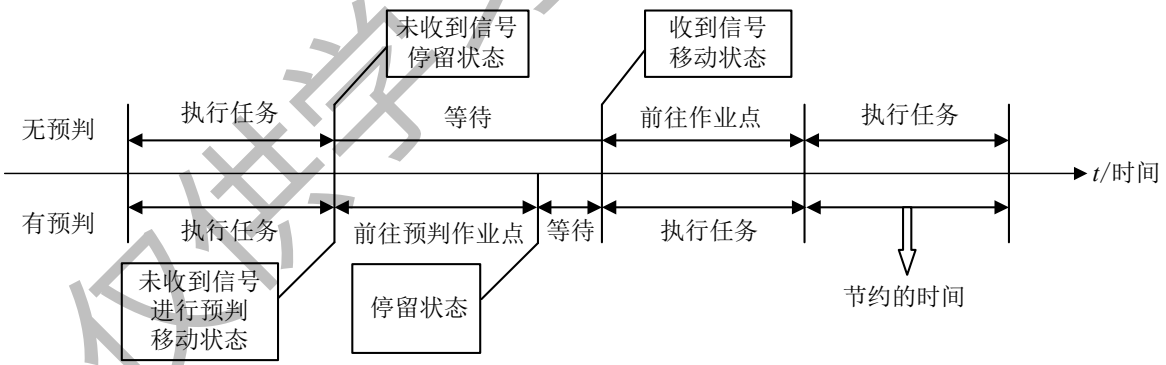


图 4-6 系统有无预判的生产时间对比

4.2 智能调度系统相关参数

智能加工系统是基于各机床的工作状态对 RGV 引导车进行调度，在进行动态调度的过程中涉及到两个调度主体——RGV 引导车、CNC 机床，模型中需要对这两个主体的状态进行合理的分析才能构建出合理的调度模型，因此我们以 RGV 引导车和 CNC 机床为中心，分别分析在动态调度过程中会对调度结果产生影响的变量，并将变量进行参数化处理以建立数学模型。

4.2.1 RGV 引导车的相关参数

由于 RGV 引导车同一时间只能执行一项动作(等待、左右运动、上下料、清洗作业)，并且只有移动到 CNC 机床前才能执行上下料、清洗物料操作。因此将 RGV 引导车的行为作为分析的主体，将其运动抽象为离散的状态转移，而对状态转移的中间过程不予以分析。

(1) RGV 引导车“停止等待”

RGV 引导车处于“停止等待”状态时，可以对需求指令及时做出响应，移动到相应的 CNC 机床，而非“停止等待”状态时，CNC 机床发出需求指令不会对 RGV 引导车的状态产生影响，RGV 引导车仍会执行已接受的指令对应的工作任务，直到任务完成才会响应需求指令。

在进行智能调度时，若在 RGV 引导车的“非停止等待”状态中进行运算，RGV 引导车并不能及时响应该指令，且在“非停止等待”状态中随时会产生新的需求指令影响调度结果，因此在 RGV 引导车“非等待状态”时进行计算将会造成计算资源浪费等情况，在模型中认为只有 RGV 引导车处于“停止等待”状态时，其调度指令才能执行，RGV 引导车的动态调度可以不必考虑 RGV 引导车“非停止等待”状态中的情形。

(2) RGV 引导车移动

RGV 引导车移动时可以往左或往右运动，规定 $\mathrm{CNC}_1 \rightarrow \mathrm{CNC}_7$ 方向(往右运动)为正方向，$\mathrm{CNC}_7 \rightarrow \mathrm{CNC}_1$ 方向(往左运动)为负方向。则 RGV 引导车在 t 时刻接受“向左/右移动 i 个单位”指令时，下一个 RGV 引导车的“停止等待”状态对应的时间点为：

$$t \leftarrow t + M_i \tag{2}$$

注 M_i 为 RGV 引导车移动 i 个单位所需的时间，补充定义 $M_0 = 0$

在公式(2)中对应的时间点时，引导车所处的位置为：

$$I_{t+M_i} = \begin{cases} I_t + i & ,\text{RGV 引导车向右移动时} \\ I_t - i & ,\text{RGV 引导车向左移动时} \end{cases} \tag{3}$$

显然，RGV 引导车的运动应在工作区域内，即：

$$\forall I_t, I_t = 0,1,2,3 \tag{4}$$

(3) RGV 引导车上下料、清洗工作

在问题分析中已经指出，加工作业所需工序的不同及各种参数的取值差异会对整个调度过程产生影响。因此接下来的讨论均针对满足以下条件的一般情况进行：

- CNC 机床加工生料的用时远大于 RGV 引导车移动的用时；
- 为同一位置奇数编号机床上料与偶数编号机床上料的用时差远小于引导车移动的用时。

此时，只需要依照加工工序数进行分类，对 RGV 引导车进行上下料、清洗工作的任务细节进行讨论。

1) 一道工序的物料加工作业情况

开始执行班次作业时，RGV 引导车位于 $I_{t=0}=0$ 位置，第一次在奇数编号机床上料结束后，下一次上料时有两种可能的选择：

- 在相同位置偶数编号机床上料
- 移动 1 单位后在奇数编号机床上料

即需要计算两者上料时间的最小值进行下一次任务的决策，具体表达式为：

$$\min\left(T_2,\ T_3+M_1\right) \tag{5}$$

由于模型假设中认为同一位置奇偶数编号机床上下料的用时差远小于移动引导车的用时，因此选择第一种方案可以实现相邻两次上料时间最短，且若该假设不满足，则每次上料完成后 RGV 引导车均会选择奇数编号引导车作为下一次任务目标，偶数编号引导车会一直处于空闲状态，不满足实际情况，该假设合理。

将上述思路整理，得到该情况下，RGV 引导车最短用时上料方案：

Step 1 RGV 引导车从上料传送带抓取生料，对当前位置奇数编号机床进行上料，完成后再从上料传送带抓取生料，对当前位置偶数编号机床进行上料，在本轮过程中不会产生清洗时间；

Step 2 若此时 CNC_1 机床未完成上料任务，则小车向右前进一步；

Step 3 重复执行 Step1 和 Step2 直到小车到达位置 3，否则 RGV 引导车返回 CNC_1 和 CNC_2 正中间(即回到初始位置，此时完成了第一轮上料)；

Step 4 从上料传送带抓取生料，再对当前位置奇数编号机床进行下料、上料(同时进行)，随后对该已加工熟料进行清洗成料，再放置到下料传送带。对当前位置偶数编号机床也进行同样的操作；

Step 5 执行路径调度算法(本文 4.3 部分)

Step 6 达到班次工作时间上限，RGV 引导车不再运动。

2) 两道工序的物料加工作业情况

开始执行班次作业时，RGV 引导车位于 $I_{t=0}=0$ 位置，由于两道工序加工任务分别在不同的机床上进行，因此其调度算法与机床类型的数量及其位置分布高度相关。如若仅有一台机床进行第一道工序的加工，则会出现多个机床空闲等待加工的情况，再者，若仅有的一台机床位于最边缘的位置，则会出现物料搬运时间耗费过多的情况，对于此类问题，在之后会建立相关的模型进行考虑，在这里先不予赘述。此处仿照第 1) 部分思路仍然可以得到完成第一轮上料的最短用时方案：

Step 1 RGV 引导车移动到距离最近的第一道工序 CNC_i 机床前；

Step 2 此时 RGV 引导车从上料传送带抓取生料，对 Step1 的机床进行上料，此时 Step1 上还未有物料，不需要进行下料、清洗；

Step 3 若还有未上料的第一道工序机床，则重复执行 Step1 和 Step2；

Step 4 当所有第一道工序机床上料结束，小车前往最早结束加工任务的第一道工序机床，等待下料、上料、清洗成料；

Step 5 执行路径调度算法(本文 4.3 部分)

Step 6 达到班次工作时间上限，RGV 引导车不再运动。

在中间的每一步，RGV 引导车在对第一道工序的机床执行任务时都需要进行抓取生料、下料、上料、移动到第二道工序机床进行下料(若机床上有加工完成的熟料)、上料、清洗熟料、放置成料等操作。

由于物料在不同工序上的加工时间不同，而清洗熟料、将物料转移的过程会导致 RGV 引导车接受到越来越多的“第一道工序任务完成信号”、“第二道工序任务完成信号”。因此，考虑到不同工序的约束，对 RGV 引导车的工作任务进行合理的决策将是智能调度的关键。

4.2.2 CNC 机床的相关参数

CNC 机床只会存在正在工作和停止等待两种状态，而 CNC 机床的工作状态会直接影响到 RGV 引导车的调度方案。因此，我们以 CNC 机床的工作状态为出发点，建立与 RGV 引导车调度过程相关的参数。

(1) CNC 机床的工作状态

定义 1 $w_{i,t}$ 表示 CNC_i 机床在 t 时刻当前执行任务的剩余时间；若机床当前未执行加工任务或加工任务已完成，则 $w_{i,t}=0$；若机床发生故障，则以机床的维修时间 F_r 替代机床的当前加工任务的剩余时间：

$$w_{i,t} = F_r \tag{6}$$

(2) CNC 机床工作状态变化

1) RGV 引导车上料作业

只有当机床的状态码 $\mathrm{CNC}_{i,t} = 0$ 时，CNC_i 机床未执行任务或当前任务已执行完成，此时该 CNC_i 机床会向 RGV 引导车发送需求信号。在 t 时刻，若 RGV 引导车为第 i 台机床安排第 j 道工序任务，则 CNC_i 机床完成当前加工任务的剩余时间 $w_{i,t}$ 为：

$$w_{i,t} = f_j \tag{7}$$

注 当前班次执行一道工序的物料加工作业时，设定 $f_j = f_0$

2) CNC 机床工作状态随时间的变化

$w_{i,t}$ 表示 CNC_i 机床在 t 时刻完成当前加工任务的剩余时间。自 RGV 引导车上料作业开始，初始状态为公式(4)，$w_{i,t}$ 开始随时间 t 变化而递减，表示机床正在执行任务。在 $t+1$ 时刻，机床完成当前加工任务的剩余时间 $w_{i,t+1}$ 按照如下规则进行更新：

$$w_{i,t+1} = \begin{cases} w_{i,t} - 1 & , w_{i,t} \neq 0 \\ 0 & , w_{i,t} = 0 \end{cases} \tag{8}$$

注 $w_{i,t}$ 表示 CNC_i 机床在 t 时刻完成当前加工任务的剩余时间

3) CNC 机床发生故障

若机床在 t 时刻发生故障，根据题设及定义 1，在此后的 F_r 时间内机床都将进行人工维修。此时 CNC_i 机床完成当前加工任务的剩余时间为：

$$w_{i,t+1} = F_r \tag{9}$$

对该机床进行维修时，将维修工作也视为机床的加工任务，仍以公式(8)对机床的剩余加工时间进行更新，当 $w_{i,\,t}$ 变为零时，则视为 CNC_i 机床维修完成。

4.3 动态规划最短路径模型

在 t 时刻，若 RGV 引导车处于“停止等待”或下一秒处于“停止等待”状态，则此时可以对 RGV 引导车进行路径规划，并通过调度系统下达指令让 RGV 引导车即刻前往下一个执行任务的位置进行工作。

若只考虑下一个即将完成工作的机床，将 RGV 引导车提前进行调度，则很有可能会陷入“局部最优”。因此，本文采用预判 k 个最先完成加工任务的机床，结合动态规划，生成若干个可能的候选路径，在其中寻找最短路径，将最短路径的起始位置作为 RGV 引导车调度的目标位置，在每一次 RGV 引导车完成任务后即重复前面的步骤进行调度。这种考虑很大程度上可以避免落入“局部最优”。

4.3.1 单工序的路径规划模型

(1) 构建决策对象

定义 2 定义 $I(\mathrm{CNC}_i)$ 为 CNC_i 车床所在位置：

$$I(CNC_i)=\begin{cases}\dfrac{i-1}{2} & ,i\text{为奇数}\\ \dfrac{i}{2}-1 & ,i\text{为偶数}\end{cases} \tag{10}$$

定义 3 在 t 时刻，各机床完成当前加工任务的剩余时间分别为 $w_{1,t},w_{2,t},\cdots,w_{8,t}$，取其中数值最小的 k 个(最快完成当前加工任务的 k 个)机床构成候选集 S_t^k。

$$S_t^k=\{CNC_i \mid w_{i,t}\in \min_k\{w_{1,t},w_{2,t},\cdots,w_{8,t}\}\} \tag{11}$$

定义 4 式中 $\min_k\{w_{1,t},w_{2,t},\cdots,w_{8,t}\}$ 表示前 k 个最小值构成的集合(含重复值)。若这 k 个时间都相同，说明在某个时间点会有多个机床同时完成任务。

定义 5 在 t 时刻的候选集 S_t^k，候选集中各机床任意排列，按照顺序组成 $k!$ 条不同的候选路径，这些候选路径构成候选路径集合 R_t^k，假定 RGV 引导车沿着各条候选路径均可以进行加工任务。

定义 6 t 时刻 RGV 引导车选择候选路径集合的第 i 条路径记为 r_i ($r_i\in R_t^k$)，并记 r_{ij} 为 r_i 路径中的第 j 个机床。

本模型的决策对象即为这些候选路径。

(2) 构建决策目标及状态方程

根据模型分析，决策目标即为选择一条候选路径使得 RGV 引导车完成这些机床任务所需时间最短。最短路径的第一个机床即为 RGV 引导车的下一个作业点。决策目标的相关表达式含义如表 4-1 所示：

表 4-1 决策目标模型相关表达式含义

表达式	含义
$\varphi(r_i)$	目标函数计算式。RGV 引导车按照候选路径 r_i 完成任务所需时间。
r_{ij}	候选路径 r_i 中的第 j 个机床

表达式	含义
α_j	RGV 引导车移动到决策路径第 j 个位置所需的时间
β_j	RGV 引导车在第 j 个地点等待并进行作业的总时间
$\mid I(r_{ij})-I_t \mid$	RGV 引导车从 I_t 位置移动到 r_{ij} 机床所在位置需要移动的单位
$M_{\mid I(r_{ij})-I_t \mid}$	RGV 引导车移动 $\mid I(r_{ij})-I_t \mid$ 单位所需的时间
$t+\alpha_j$	RGV 引导车移动到第 j 个地点时的时刻
$w_{r_{ij},t+\alpha_j}$	$CNC_{r_{ij}}$ 机床在 $t+\alpha_j$ 时刻完成当前加工任务的剩余时间(公式(6))
$T_{r_{ij}}$	r_{ij} 机床一次上下料所需时间
T_w	熟料清洗及运出时间

在 t 时刻，RGV 引导车从 I_t 位置出发前往 r_{ij} 机床所在位置所需时间：

$$\alpha_j = M_{|I(r_{ij})-I_t|} \tag{12}$$

随后在 r_{ij} 机床处等待并进行作业的总时间为(包括进行上下料、清洗熟料、送出成料)：

$$\beta_j = w_{r_{ij},t+\alpha_j} + T_{r_{ij}} + T_w \tag{13}$$

此时，系统运行时间更新为：

$$t \leftarrow t+\alpha_j+\beta_j \tag{14}$$

时间更新时，相关参数也会更新，公式(12)~(14)即为完整的第 j 阶段的过程。从 $j=1$ 开始进行迭代，可以得到目标函数的表达式为：

$$\varphi(r_i) = \sum_{i=1}^{k}(\alpha_i+\beta_i) \tag{15}$$

(3) 动态决策流程

对于每一个 RGV 引导车的“停止等待”状态或下一秒进入“停止等待”状态都按照上述模型对下一个作业点进行预判。而预判的目标是使得：

$$\min \varphi(r_i) \tag{16}$$

式(18)计算得到的最优解在很多情况下是不唯一的。如以附件 1 第 1 组参数为例，系统刚开始作业时，第一次的路径规划时得到如表 4-2 所示的情况：

表 4-2 路径规划及耗时情况分析

候选路径	完成路径总耗时	是否最优
CNC_1-CNC_2-CNC_3	107	是
CNC_1-CNC_3-CNC_2	127	否
CNC_2-CNC_1-CNC_3	107	是

候选路径	完成路径总耗时	是否最优
CNC_2-CNC_3-CNC_1	127	否
CNC_3-CNC_1-CNC_2	127	否
CNC_3-CNC_2-CNC_1	127	否

针对这种存在多个达到最优的情形指定如下规则：

- 若各最优解区别仅为奇偶数编号前后上下料顺序不同(表 4-2 的第一条路径和第三条路径)，则优先对奇数编号进行作业。

这样，RGV 引导车便会以 CNC_1-CNC_2-CNC_3 作为最后的路径，从而该动态规划的结论为：RGV 引导车下一个工作地点为 CNC_1 机床。

当 RGV 引导车对 CNC_1 机床的上下料及清洗熟料作业完成后，RGV 引导车再次处于空闲状态，重新进行动态规划。

综合以上分析，绘制出单工序物料加工情况下的动态规划最短路径的模型求解流程图：

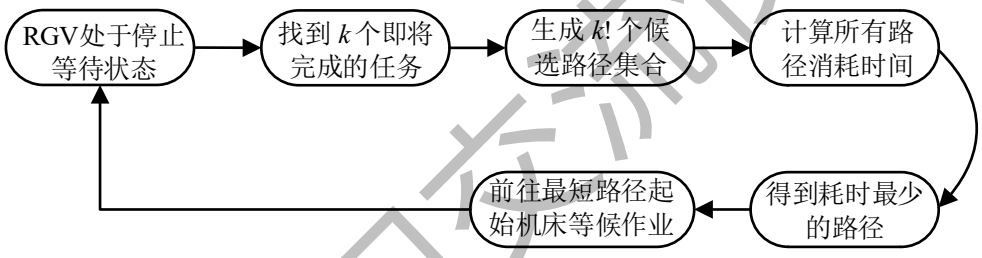


图 4-7 单工序情况下动态规划最短路径流程图

4.3.2 执行两道工序的路径规划模型

(1) 路径规划影响因素确定

1) CNC 机床类型及分布差异

考虑两道工序的物料加工情况，本文假设在一个班次作业时一台 CNC 机床只负责一道工序(不进行换刀)，先考虑如下两种极端情形：

- 每一台 CNC 机床只负责第一道工序或第二道工序；
- CNC_1 机床和 CNC_2 机床负责第一道工序，其余机床负责第二道工序；

在第一种情况中，由于物料无法完成两道工序，因此成料的产出量为 0，作业系统效率最低；在第二种情况中，两台机床都位于同一位置，在每一台负责第二道工序的机床完成任务后，机床都要回到 0 位置重新下料、上料、搬运半熟料，物料搬运时 RGV 引导车的移动过程占用了较多的时间，若将此类机床放置在作业流水线中央则可以有效减少 RGV 引导车移动过程占用的时间。

因此，合理分配执行不同工序的机床数量及其位置排布对规划路径的性能有极其重要的影响。

2) RGV 引导车工作路径探讨

RGV 引导车在不同机床间进行调度时，若不能合理地规划路径，会有大量的时间用于等待及重复地移动。考虑两道工序的物料加工情况，本文假设在一个班次作业时一台 CNC 机床只负责一道工序(不进行换刀)，则完成一个物料的加工至少需要经过以下几个步骤：

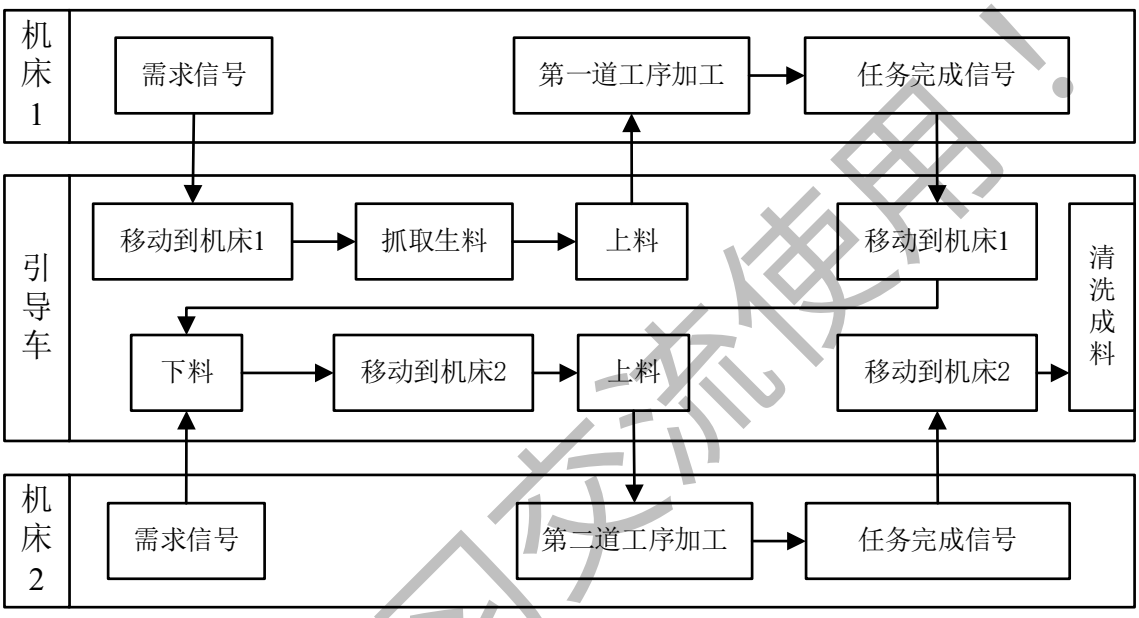


图 4-8 两道工序物料加工作业加工成料的基本步骤

每一个生料第一道工序加工结束时，RGV 引导车需要移动到该机床前，将该半熟料下料、搬运至第二道工序加工的机床处。在下料过程中，不论是否为机床上料，其消耗的时间都相同，因此为达到机床利用率最大化，应该保证在每一次下料时都进行上料；每一个半熟料第二道工序加工结束时，RGV 引导车需要移动到该机床前，将该熟料下料、清洗成料，若此时需要上料，则 RGV 引导车上一次执行任务点需要为第一道工序机床(此时才有可能获得用于第二道工序加工的半熟料)。

因此，在所有加工第一道工序的机床都开始作业时(或当前班次已经作业过至少 1 次)，针对作业点类型先后的不同可以总结出以下两种情形：

① 为了使资源充分利用，RGV 引导车在第一道工序机床进行下料时一定会进行上料(从上料传送带抓取生料)，并且下一次作业点类型为第二道工序机床。

分析 若下一次作业点类型为第一道工序机床，则 RGV 引导车不能抓取生料(否则一面为生料，另一面为半熟料，无法进行交换)，只能直接下料。此时，RGV 引导车上携带两块半熟料，需要先移动到第二道工序机床上没有物料的机器进行上料。这种情况

下，虽然仍能生产成料，但是造成了两台机床空闲，没有合理利用资源，也可知并非最优选择方案。

② 若 t 时刻 RGV 引导车在第二道工序的机床进行上下料，则下一次作业时可以选择第一道工序的机床，也可以选择第二道工序机床。但为了保证下一次的成料加工效率最大，应该选择第一道工序机床。

分析 若选择第二道工序机床，由于 RGV 引导车上没有携带半熟料，只能对第二道工序机床进行下料、清洗。这种情况下，由于 RGV 引导车每次到达机床时都会重复进行下料、上料、清洗的工作，先对第二道工序的机床进行下料和清洗并不会节省任何时间，因为总是需要 RGV 引导车带着加工完的第一道工序的物料再重复同样的过程，且可能会出现耽误时间的情况，如在对第二道工序机床进行工作时有第一道工序机床已经完成加工。在这种情况下进行规划，会造成计算的冗余，却不会对结果产生任何优化，甚至会出现增加时间的情况，因此在之后不予考虑。

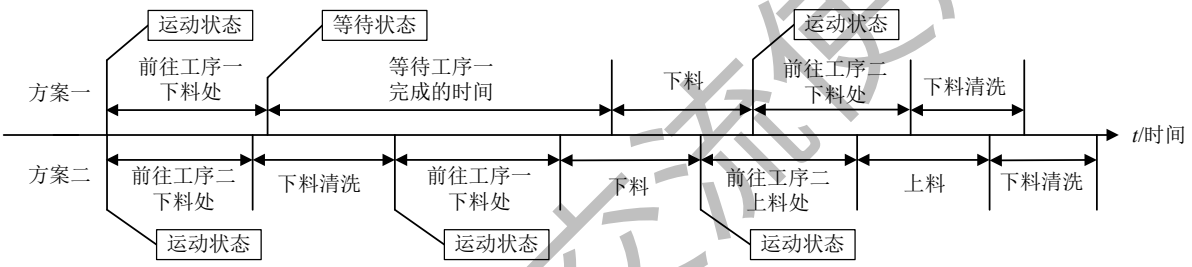


图 4-9 上述一、二方案时间对比图

进一步分析，在机床类型数量及分布确定的情况下，作业系统只需要根据需求信号的先后顺序执行作业任务，就能保证最基本的执行情况。智能调度方案可以优化 RGV 引导车的作业次序，从而提高效率。

因此，①部分决定了作业系统的下限，②部分决定了作业系统的上限。对于两道工序的机床加工任务，其路径规划调度需要考虑以下三点：

- 确定合理的机床类型数量/比例；
- 不同类型的机床在流水线上的排列；
- 使得加工第二道工序的机床资源利用率最大化。

(2) 两类工序机床数量确定

两类工序机床比例与两道工序的加工时间密切相关，如：加工第一道工序所用时间比第二道工序时间长，则应该安排较多的第一道工序类型机床。

考虑在稳定情况下，两类工序机床的比例应该为加工时间之比，但由于加工时间之比不一定是整数，且两类机床数量不相等时，物料的转运过程也会占用一定的时

间。因此，设 N_1 台机床加工第一道工序，N_2 台机床加工第二道工序，则合适的 N_1 和 N_2 应满足如下关系式：

$$\begin{cases} N_1+N_2=8 \\ N_1,N_2\text{是正整数} \\ \dfrac{f_1}{f_2}\in\left[\dfrac{N_1-1}{N_2+1},\dfrac{N_1+1}{N_2-1}\right] \end{cases} \tag{17}$$

以附件 1 表 1 中的三组参数为例，计算每组中较为合适工序机床数量如表 4-3 所示：

表 4-3 附件 1 表 1 计算出较合适的工序机床数量

组数	f_1	f_2	f_1/f_2	两类工序机床数量			
				N_1	N_2	N_1	N_2
第 1 组	400	378	1.0582	4	4	5	3
第 2 组	280	500	0.5600	2	6	3	5
第 3 组	455	182	2.5000	5	3	6	2

(3) 动态规划最短路径模型改进

该作业系统中，第二道工序机床的工作效率将直接影响成料产量。结合上文对工作路径的探讨，可以得到 RGV 引导车的工作路径应该为第一道工序、第二道工序交替进行。为充分调用机床资源并提高成料产量，将 4.3.1 的模型扩展为 1-2-1 调度模型和 2-1-2 调度模型。针对不同工序机床数量/比例，从这两种调度模型中选择效果较好的模型。

定义 7 执行第一道工序的机床构成的集合为 U_1，执行第二道工序的机床构成的集合为 U_2

定义 8 U_1 中完成当前加工任务剩余时间最短的两个机床分别为 u_{11},u_{12}，且有$u_{11}\le u_{12}$；U_2 中完成当前加工任务剩余时间最短的两个机床分别为 u_{21},u_{22}，且有$u_{21}\le u_{22}$

1) 1-2-1 调度模型

4.3.1 构建的单工序路径规划模型可以看作 1-1-1 调度模型，即选择 3 个第一道工序的机床，使得完成该路径的用时最短。而 1-2-1 调度模型可以看做是以加工完成尽可能多的第一道工序物料为规划目标，视为在一个第一道工序的机床和另一个第一道工序的机床之间插入一个第二道工序的机床，使得完成该路径的用时最短。即在 U_2 中选择一个第二道工序机床，使得完成路径 $u_{11}\to?\to u_{12}$ 的总用时最少。

2) 2-1-2 调度模型

与 1-2-1 调度模型类似，在 2-1-2 调度模型中可以看做是以加工完成尽可能多的第二道工序物料为规划目标，视为在一个第二道工序的机床和另一个第二道工序的机床之间插入一个第一道工序的机床，使得完成该路径的用时最短。即在 U_1 中选择一个第一道工序机床，使得完成路径 $u_{21} \to ? \to u_{22}$ 的总用时最少。

将这些路径作为 4.3.1 中的候选路径集合 R_t^k，应用该动态规划模型决策出最优的路径。

在迭代进行路径规划时，得到的最优路径中的路径起点是上一阶段决策路径的终点，因此，对每一条决策出来的路径，RGV 引导车会对最优路径中的前两台机床进行作业任务，随后 RGV 引导车再次处于空闲状态，重新进行动态规划。

(4) 基于单亲遗传算法的最优分布模型

由(2)确定了两类工序机床的数量分别为 N_1 和 N_2，则共有 $C_8^{N_1}$ 种可能的分布情况。当总机床数较少时，可以枚举每一种可能的分布情况并计算一个班次内该机床安排方案下的最优加工物料总数，即可选出较优的安排方案。但当总机床数增多、工序增多时，枚举方法便不再适用了，因此本文提出使用单亲遗传算法进行最优分布规划。单亲遗传算法(简称 SAPGA)的介绍见本文附录 A。

下面介绍单亲遗传算法在工序机床最优分布中的应用：

编码 对于每一个个体 X，每一个维度上的编码根据其所在的取值区间确定了该机床的工作类型。若 $x_i \in \{1,2,\cdots,N_1\}$，表明第 CNC_i 机床为第一道工序机床，否则为第二道工序机床。

适应度 SAPGA 的适应度即为在当前安排方案下，一个班次的机床加工作业最多能加工的成料个数，适应度越大表示分布方案越好。

算法开始时，会随机生成初始值(初始分配方案)。此时对这些分配方案进行仿真调度，计算出适应度函数值(成料个数)，选择较好的分配方案予以保留，较好的分配方案有一定概率可以发生变异和对换，从而产生新的解。重复进行上述过程，直到达到终止条件时算法会返回一个经过多代繁衍稳定下来的最优解，该最优解代表的分配方案即为最优的分配方案。

4.3.3 机床发生故障时路径重新调配问题的简要分析

在 4.2.2 中已经指出，机床发生故障时，将机床的剩余维修时间看做机床的剩余工作时间，当机床维修结束后可以再加入作业列表时。在两道工序的情形中，新机床第一次上料时没有物料产出，可能会对调度产生影响。因此，机床维修结束后是否继续工作，需要对其影响进行权衡。

由定义 1，CNC_i 机床发生故障时，该机床的当前加工任务的剩余时间 $w_{i,t}$ 更新为机床维修时间 F_r。维修初期，该机床的 $w_{i,t}$ 较大，路径规划时不会将其作为候选作业点；维修末期，该机床的 $w_{i,t}$ 较小，将会被选为候选作业点。

(1) 单工序机床发生故障

单工序机床加工发生故障时，由于本文的动态规划算法结合了预判机制，可以在很大程度上缓解机床故障维修期间对生产造成的影响。在实际进行决策的过程中将故障机床从决策对象中剔除即可，当维修完成时，再将其加入列表进行考虑。因此，考虑故障发生的情况，只需要对多工序机床的模型算法进行进一步优化。

(2) 多工序机床发生故障

多工序机床发生故障时，会对 CNC 机床的类型比例及分布差异造成影响，从而影响整体作业系统的加工效率。

- 作业系统未有机床发生故障时，对该调度系统使用 4.3.2 建立的模型进行调度
- 在 t 时刻，作业系统开始有机床发生故障时，若故障发生的机床与当前的候选路径不冲突，则按照当前的路径规划进行调度，否则先进行计算是否需要调整调度模式，再进行调度。
- 在 t 时刻，RGV 引导车接受调度指令，前往下一个作业点或正在等待作业点完成加工任务时，若该作业机床发生故障，则 RGV 引导车在最近的“停止等待”状态时先进行计算是否需要调整调度模式，再重新规划路径。

此时将问题转化为在两道工序机床数量比例与两道工序加工时间比例失衡时，如何进行调度的问题。缓解这种资料调配不均的影响，需要根据当前的作业情况，及时调整当前的调度模型(1-2-1 模式还是 2-1-2 模式)。

五、模型求解

针对一道工序的物料加工作业情况，结合建立的动态规划最短路径的模型，在每次对 RGV 引导车进行调度前，选择当前时刻最快结束任务的 3 个机床进行最短路径的规划，运用动态规划算法对问题进行求解；

针对两道工序的物料加工作业情况，首先要考虑负责两道工序的机床数量与位置分配的问题，我们采取改进后的单亲遗传算法这一部分进行求解。之后再利用改进后的的动态规划最短路径模型对问题进行求解，在这个基础上，我们考虑了两种模型：1-2-1 模型与 2-1-2 模型，这两种模型分别适用于不同类型的情况，因此我们运用两种模型分别对三组数据进行求解，以得到更优的结果；

针对故障可能发生的情形，只需要在前两种情形的基础上忽略故障机床进行求解即可。

5.1 一道工序加工作业情况的调度策略

将题目所给的三组系统作业参数代入模型中，运行算法进行求解，得到结果如表5-1 所示(只截取部分结果进行展示)：

表 5-1 一道工序时第一组数据输出的部分结果

加工物料序号	加工 CNC 机床编号	上料开始时间	下料开始时间	加工物料序号	加工 CNC 机床编号	上料开始时间	下料开始时间
1	1	0	588	382	6	28088	28679
2	2	53	644	383	7	28164	28755
3	3	129	720	384	8	28217	
4	4	182	773	385	1	28319	
5	5	258	849	386	2	28421	
6	6	311	902	387	3	28497	
7	7	387	978	388	4	28550	
8	8	440	1031	389	5	28626	
9	1	588	1176	390	6	28679	
10	2	644	1235	391	7	28755	

我们选取了第一组输出数据的起始与末尾处各 10 个输出结果进行展示，第一组数据一个班次内实际上只加工完成了 382 个物料，其中第 383 个物料无法在八小时内完成上下料与清洗的过程，认为其加工未完成，其余同理。

根据表格可以发现，RGV 引导车的调度呈现一定的周期性，即 RGV 引导车始终按照机床编号顺序依次给机床进行上下料等工作，在八台机床任务完成后再返回起始位置 CNC_1 处重复上述过程。这个过程在表中表现为加工 CNC 机床编号变化的规律性，即编号变化总是由 $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 1$ 。由于第二、三组输出结果也呈现出相同的规律特征，因此我们只对第一组数据的结果特征进行分析，其他两组类似，在这里不予赘述。

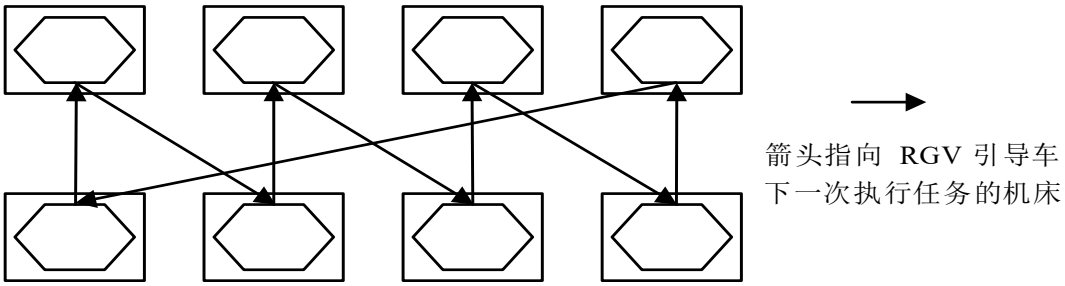


图 5-1 RGV 引导车周期性调度演示

对这种现象进行深入分析发现，在只有一道工序的物料加工过程中，不考虑问题分析中的情形一，由于第一轮 RGV 引导车进行调度时是按照机床编号顺序依次进行工作的，每台机床加工开始的时间间隔恒定不变，因此按照编号顺序对 RGV 引导车进行调度并不会造成任何多余的时间损耗(如图 5-2)。但如果按其他方式进行调度，则会由于 RGV 引导车移动时间的存在，造成机床产生等待时间，降低生产效率。

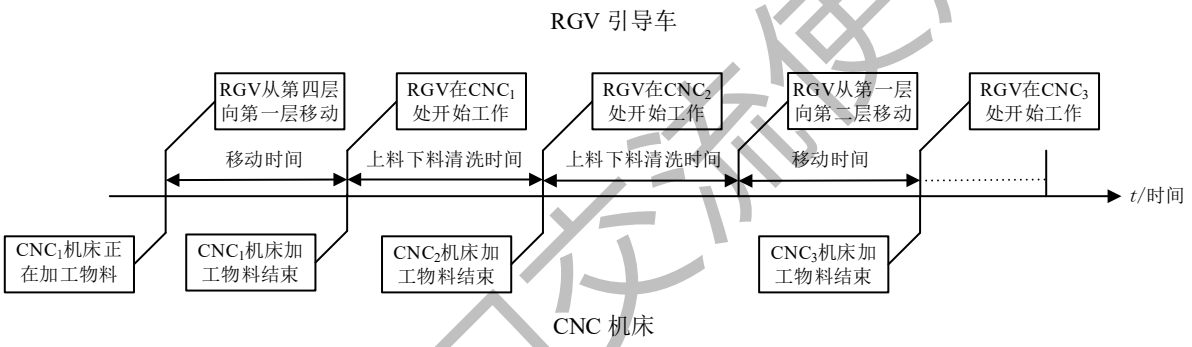


图 5-2 调度路线示意图

于是得到单工序时 RGV 引导车的调度策略为：按机床编号顺序依次进行工作，完成一台机床的工作后立即前往下一个编号的机床进行工作。

5.2 两道工序加工作业情况的调度策略

将题目所给的三组系统作业参数代入模型中，使用 Python 进行编程求解。代码直接求解结果见支撑材料，使用单亲遗传算法分别对不同的 N_1 及调度模式进行求解，求解结果如表 5-2 所示：

表 5-2 单亲遗传算法对不同模式、不同 N_1 求解结果

组别	调度模式	N_1	最优机床分布	最大零件个数
第 1 组	1-2-1 调度模式	4	[1,2,2,1,2,1,2,1]	252
		5	[2,1,1,1,2,1,1,2]	252
	2-1-2 调度模式	4	[1,2,2,1,1,2,2,1]	205
		5	[1,1,1,2,1,2,1,2]	195
第 2 组	1-2-1 调度模式	2	[1,2,2,2,1,2,2,2]	180

		3	[2,2,2,1,1,2,1,2]	197
	2-1-2 调度模式	2	[1,2,2,2,1,2,2,2]	166
		3	[1,2,1,2,1,2,2,2]	171
第 3 组	1-2-1 调度模式	5	[1,1,2,1,2,1,1,2]	239
		6	[1,1,2,1,2,1,1,1]	233
	2-1-2 调度模式	5	[1,1,2,1,2,1,1,2]	199
		6	[1,1,2,1,2,1,1,1]	199

在上表中发现，1-2-1 调度模型的调度效果均优于 2-1-2 模式。其可能的原因是，以第二道工序机床的调度顺序优先的话，可能会造成半熟料的产能不足，从而限制了生产效率。

选取第一组的部分结果如表 5-3 所示：

表 5-3 两道工序加工作业情况(选取部分)

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	3	527	763
2	2	27	597	5	661	870
3	4	77	731	3	763	999
4	6	127	833	5	870	1128
5	7	177	940	3	999	1262
6	1	482	1069	5	1128	1364
7	2	597	1212	3	1262	1582
8	4	731	1314	5	1364	1711
9	6	833	1416	8	1466	2049

这里，我们只选取了第一组输出数据的起始处的 9 个输出结果进行展示，后面的 RGV 引导车的调度过程与展示的数据类似，因此只对列出的数据进行分析说明。

根据表格可以发现，RGV 引导车的调度呈现一定的周期性，即 RGV 引导车的作业顺序为：$1\rightarrow 2\rightarrow 4\rightarrow 6\rightarrow 7\rightarrow 1\rightarrow 3\rightarrow 2\rightarrow 5\rightarrow 4\rightarrow 3\rightarrow 6\rightarrow 5\rightarrow 7$，其中编号为 1、2、4、6、7 的机床负责进行第一道加工，编号为 3、5 的机床负责进行第二道加工。可以发现，在 RGV 引导车第一轮调度过程中，会依次按编号顺序给负责第一道工序加工的机床进行上料，接着随着半熟料加工完成，RGV 引导车会按照到达当前时刻最快结束的负责第一道加工任务的机床处进行上下料，将其运送到距离最近的负责第二道加工任务的机床处进行上下料、清洗，完成任务后再重复上述操作的过程进行调度。其他两组数据也呈现类似的调度结果，在这里不予赘述。

系统调度的规律性结果与一道工序的加工作业情形类似，因此在这里不再进行深入分析。于是得到两道工序时 RGV 引导车的调度策略为：第一轮调度过程中，按机床

编号顺序给负责第一道工序加工的机床进行上料，将 RGV 引导车调度到当前时刻最快结束的负责第一道加工任务的机床处进行工作，接着调度到距离最近的负责第二道加工任务的机床处进行工作，完成任务后再重复上述操作的过程进行调度。

5.3 一、二道工序加工作业情况的工作效率

为衡量系统的作业效率，我们选取一个班次内的机床空闲平均时间作为衡量指标。若一个班次内机床的平均空闲时间少，每时刻空闲的机床数少，则说明整个加工系统的作业效率高，模型实用性高，可以用于实际的生产生活。

为了计算一个班次内的机床空闲平均时间，在模型建立中已经定义了 $w_{i,t}$ 为 CNC_i 机床 t 时刻当前执行任务的剩余时间，对于每一个机床，搜寻每一次剩余时间即 $w_{i,t}$ 从非零变到零的过程，记录剩余时间为零的时刻为初始时刻，接着再搜寻下一次剩余时间即 $w_{i,t}$ 从零变到非零的过程，将非零时刻作为末尾时刻，作差即得到一次工作的上料时间+工作时间+等待时间，继续减去上料时间+工作时间即得到了等待时间。我们利用输出的数据结果进行计算，即可直接得到所需的机床平均等待时间。

表 5-4 根据附件数据计算得到的机床平均等待时间

		加工零件总个数	机床平均等待时间
附件 1	第 1 组	382	3.141s
	第 2 组	359	17.433s
	第 3 组	392	4.081s
附件 2	第 1 组	252	23.927s
	第 2 组	197	139.943s
	第 3 组	239	117.55s

附件一对应一道工序的物料加工情况，附件二对应两道工序的物料加工情况。可以看到，在一个班次八小时内的系统工作时间中，最长的机床平均等待时间仅为 140s，不到三分钟的时间。因此，可以从表格中看出系统的工作效率很高，机床几乎每时每刻都在进行工作，资源得到了十分有效的利用，同时，也说明模型的实用性很高，可以引入实际的生产生活中。

5.4 基于故障考虑的一、两道工序加工作业情况

在 5.1.1、5.2.1 中已经得到了一道工序与两道工序情形下的 RGV 引导车调度策略，因此当机床发生故障时，在动态规划的过程中只需要忽略故障的机床，对其他机床延续原有的调度策略即可。

由于故障发生的时刻与发生故障的机床具有随机性，计算出的机床平均空闲时间也具有随机性，因此在这里不单独根据机床的空闲时间进行说明，而采取更加合理的方法衡量系统的工作效率。

为了衡量系统的工作效率，我们根据题目所给的三组数据，在一道工序加工物料时对于每一次的上料，都用随机数控制是否发生故障、将在哪里发生故障；两道工序加工物料时随机选取两个机床发生故障。使用这种故障策略进行 100 个班次的随机独立重复实验，计算出每一组加工的成料个数。模型仿真结果如下图所示：

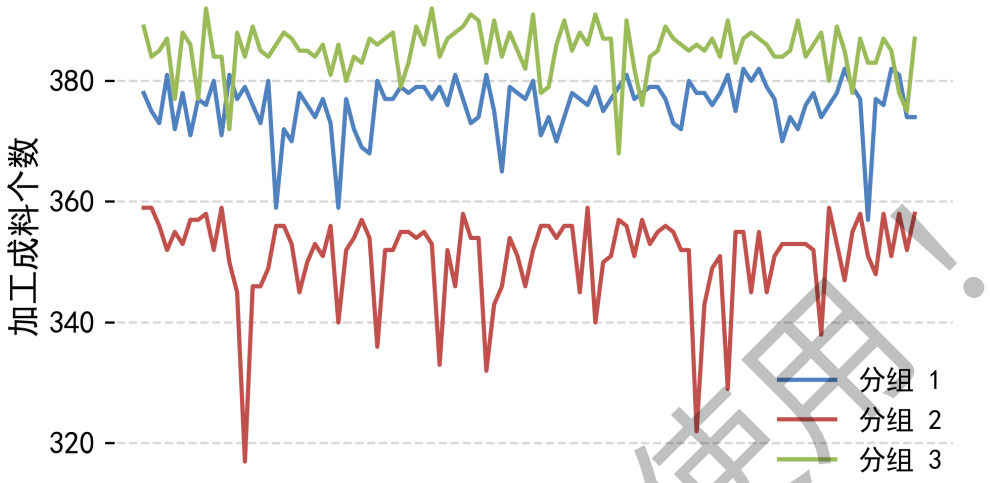


图 5-3 一道工序加工物料的成料个数仿真结果

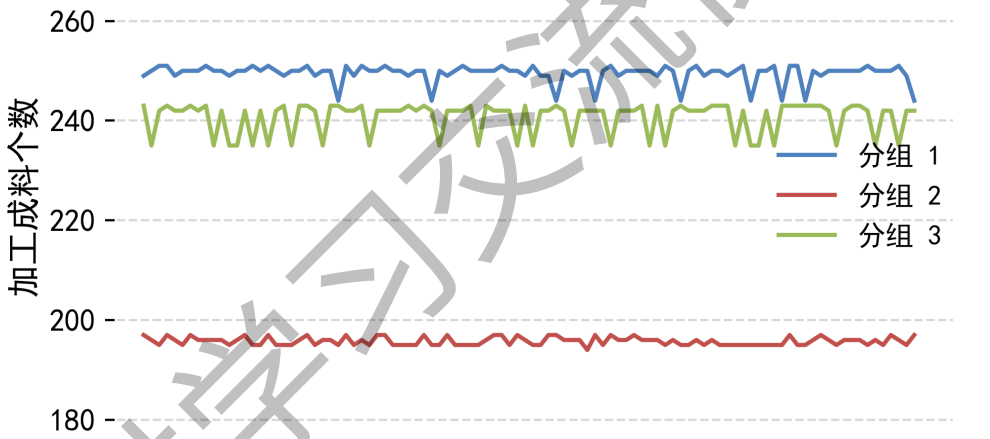


图 5-4 两道工序加工物料的成料个数仿真结果

由图 5-3、图 5-4 可看到，在次 100 班次实验中成料加工个数在一定区域内波动，不会产生较大的差别，模型具有很好的稳定性，也就是说，系统中一台机床发生故障，并不会对整个加工系统造成较大的干扰，与无故障情形的成料加工个数进行对比发现，故障不会对结果产生较大的影响，系统仍然保持较高的工作效率。

六、模型评价与推广

6.1 模型的优点

(1) 智能加工系统的调度过程大都以图像和表格的形式呈现，直观地反映了模型分析、建立与求解的过程，便于读者阅读和理解；

(2) 与传统调度系统相比，本文的智能加工系统加入预测能力，从而节省大量等待时间，提高了系统的效率；
(3) 模型运用排队论与决策论思想，合理设置 RGV 引导车和 CNC 机床的参数，使智能加工系统合理且高效的调度；
(4) 模型思路清晰，简单易懂，算法简单易实现，同时与实际没有产生很大的误差；
(5) 充分结合实际，考虑各种可能情况，使模型具有高度实用性与抗干扰性，可引入实际的生产生活。

6.2 模型的缺点

(1) 在两道加工工序的调度中我们只能得到局部最优调度很难得到整体最优调度；
(2) 由于时间限制，没有对模型结果进行检验与误差分析;
(3) 参考文献[1]中指出数控机床故障系统的间隔时间服从的分布为指数分布或韦布尔分布，而本文仅仅做随机重复实验进行拟合，可靠性不足。

6.3 模型的改进

(1) 在智能加工系统进行预测时可以增加决策集的个数使得所得结果更贴近于整体最优调度策略；
(2) 当考虑智能加工系统存在故障率时，我们可以先人为添加每台机床的运行时间，使得即使机床出现故障时仍然能完成预先给定的目标，从而达到提高系统容错率的目的；
(3) 在原有模型中对于故障处理的基础上，我们可以用均匀分布去随机生成机床出现故障的次数，然后使用本文建立的智能加工系统对结果进行仿真；
(4) 利用误差分析的结果对模型的参数进行重新设定，使得系统更为高效。

参考文献

[1] 吴茂坤. 基于可靠性分析的数控组合机床维修时间设计[D]. 吉林大学, 2016.

[2] 余锦华, 杨维权. 多元统计分析与应用[M]. 中山大学出版社, 2005.

[3] 茆诗松, 程依明, 濮晓龙, 等. 概率论与数理统计教程第二版[M]. 北京: 高等教育出版社, 2011

[4] 何晓群, 刘文卿. 实用回归分析[M]. 中国人民大学出版社, 2015.

[5] 张良均. Python 数据分析与挖掘实战[M]. 机械工业出版社, 2016.

附　录

附录 A：单亲遗传算法介绍

(1) 单亲遗传算法简介

单亲遗传算法是模拟自然界少数生物单亲繁殖方式提出的一种启发式搜索算法。基本思想与传统遗传算法类似，在生成解的阶段就使用区间不重复抽样法生成没有任何维度重复的个体；在变异、交叉阶段使用独特的算子，这样就能保证在遗传算法的每个阶段所有的解都是可行解。

1) 再生算子

为了防止在进化过程中产生的最优解被交叉算子和变异算子破坏，设定精英个体比例为 0.2。对适应度较高的个体原封不动地复制到下一代，作为子代种群的一部分。

2) 变异算子

变异算子是维持遗传算法种群多样性、防止过早陷入局部最优解的基本算法。设 P 是个体的变异概率，则对于任意解 $x=(x_1,x_2,\cdots,x_m)$，本文定义变异算子为重排列算子和轮换算子：

① 重排列算子

重排列算子定义为发生变异的个体 x 在所有维度上的解的随机重排列。记为：

$$x'=(x'_1,x'_2,\cdots,x'_m)$$

其中，$x'_1,x'_2,\cdots,x'_m$ 是 $x_1,x_2,\cdots,x_m$ 的随机排列。

② 轮换算子

轮换算子定义为将当前解按照顺序排列、首尾相连封闭成一个环，然后从环中随机选择一个位点剪开形成新的解，记为：

$$x'=(x_k,x_{k+1},\cdots,x_n,x_1,x_2,\cdots,x_{k-1})$$

其中，$k\in Z\cap[0,\ n]$

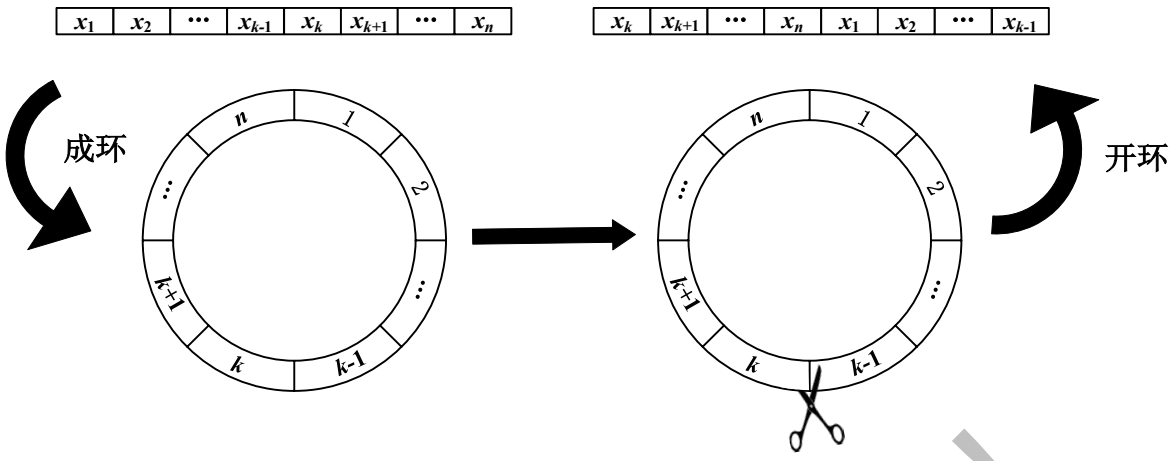
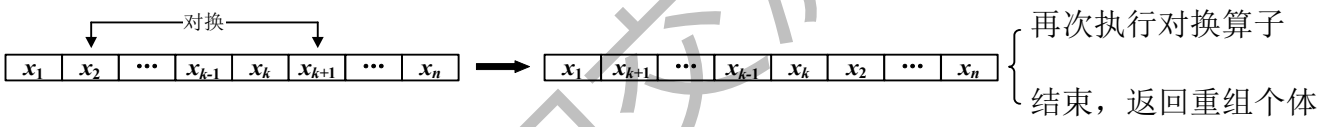


图 6-1 轮换算子示意图

3) 概率对换算子

重组算子定义为将当前解任意取出两个维度进行数值对换，形成新的解。为避免个体交换维度时出现死循环，本文引入概率对换机制，每一次对换时都有 1/2 的概率继续发生对换，也有 1/2 的概率结束对换算子，返回重组个体。理论上，发生对换的个体有：$1/2^k$ 的概率对换 k 次。



以上即为单亲遗传算法的主要内容，该算法的编程实现见代码清单 2。

附录 B：主要程序

本文代码量较大，只在论文中展示较少的部分。

代码运行环境均为 Anaconda3(python3.6)

代码清单 1　单工序的物料加工作业的动态规划求解算法

```
import os
import pandas as pd
import itertools

def I(num):
    # 车床所在位置的映射函数
    if num % 2 == 0:
        # 若为偶数
        return int(num/2 - 1)
    else:
        return int((num - 1) /2)
```

```
def w(w_it,t_pass):
    # 计算 i 机床 t 时刻经过 t_pass 时间后的任务剩余执行时间
    return w_it - t_pass if w_it - t_pass >0 else 0

def updata_w(w_its, t_pass):
    return [w(wit,t_pass) for wit in w_its]

def main(M1,M2,M3,f0,Tw,T_奇,T_偶,table,k = 3):
    # 初始化参数
    M1, M2, M3, f0, Tw = M1, M2, M3, f0, Tw
    T1 = T3 = T5 = T7 = T_奇
    T2 = T4 = T6 = T8 = T_偶

    t = 0
    k = k  # 预判 k 个机床
    I_t = 0  # RGV 所在的位置
    M0 = 0
    t_max = 8 * 60 * 60
    w_its = [0] * 8

    n = 0
    while t <= t_max:
        n += 1
        w_its_sorted = sorted(w_its)    # 对时间进行排序

        Stk = []    # 候选集
        for CNC_i in range(1,9):
            if w_its[CNC_i - 1] in w_its_sorted[:k] and len(Stk) < k:
                Stk.append(CNC_i)   # 取出前 k 个时间对应的机床编号

        Rtk = list(itertools.permutations(Stk, k))  # 候选路径
        varphi = [] # 目标函数值

        # 动态规划寻找最优路径
        for ri in Rtk:
            t_virtual = t   # 虚拟时间
            I_t_virtual = I_t   # 虚拟 RGP 地点
            j = 0
            w_its_virtual = w_its.copy()
            while j <= k - 1:
                alphaj = eval(f'M{abs(I(ri[j]) - I_t_virtual)}')
                betaj = eval(f'T{ri[j]}') + Tw + w(w_its_virtual[ri[j] - 1],
alphaj)
                t_virtual += alphaj + betaj  # 新的系统时间
                I_t_virtual = I(ri[j])    # 新的 RGV 位置
                w_its_virtual[ri[j] - 1] = f0 + alphaj + betaj - Tw  # 更新时间表
```

```
                w_its_virtual = updata_w(w_its_virtual,alphaj + betaj)
                j+=1
            varphi.append(t_virtual)

        move_index = [] # 储存最小路径代价
        for j in range(len(varphi)):
            if varphi[j] == min(varphi):
                move_index.append(Rtk[j])

        move_point = [] # 最优机床方案
        for move_index_road in move_index:
            move_point.append(move_index_road[0])
            if move_index_road[0] % 2 == 1:
                # 如果是奇数号则直接输出结果
                move_point = [move_index_road[0]]
                break

        # 确定最终结果，更新参数
        alpha1 = eval(f'M{abs(I(move_point[0]) - I_t)}')    # 移动到最优地点所需
的时间
        table.at[n,'加工 CNC 编号'] = int(move_point[0])
        table.at[n,'上料开始时间'] = t + alpha1 + w(w_its[move_point[0] - 1],
alpha1)
        print(f'CNC 机床编号: {move_point[0]}, 上料开始时间:{t + alpha1 +
w(w_its[move_point[0] - 1], alpha1)}')
        beta1 = eval(f'T{move_point[0]}') + Tw + w(w_its[move_point[0] - 1],
alpha1)
        I_t = I(move_point[0])
        t +=  alpha1 + beta1
        w_its[move_point[0] - 1] = f0 + alpha1 + beta1 - Tw
        w_its = updata_w(w_its, alpha1 + beta1)
    if t + Tw + f0 >t_max:
        print(f'运行完毕，总加工零件数为：{n - 8 - 1}')
    else:
        print(f'运行完毕，总加工零件数为：{n - 8}')

    # 填充表格
    for i in range(1,len(table)+1):
        for j in range(i + 1, len(table) + 1):
            if table.at[i,'加工 CNC 编号'] == table.at[j,'加工 CNC 编号']:
                table.at[i,'下料开始时间'] = table.at[j,'上料开始时间']
                break

if __name__ == '__main__':

    data1 = pd.DataFrame(columns=['加工 CNC 编号', '上料开始时间', '下料开始时间
```

```
'])
    data2 = pd.DataFrame(columns=['加工 CNC 编号', '上料开始时间', '下料开始时间
'])
    data3 = pd.DataFrame(columns=['加工 CNC 编号', '上料开始时间', '下料开始时间
'])

    main(20,33,46,560,25,28,31,data1)
    main(23,41,59,580,30,30,35,data2)
    main(18,32,46,545,25,27,32,data3)

    excel_writer = pd.ExcelWriter(f'{os.getcwd()}/../table/Case_1_result.xlsx')
    data1.to_excel(excel_writer, '第 1 组')
    data2.to_excel(excel_writer, '第 2 组')
    data3.to_excel(excel_writer, '第 3 组')
    excel_writer.save()
```

代码详见: code/problem1.py

代码清单 2　单亲遗传算法主程序

```
import random

class SAPGA(object):
    def __init__(self, fitness, bounds, param=tuple(), popsize=50, elite=0.2,
mutprob=0.2, maxiter=5,disp = False):
        self.fitness = fitness  # 适应度函数
        self.bounds = bounds    # 范围限制
        self.param = param      # 适应度函数的其他参数
        self.popsize = popsize  # 种群规模
        self.elite = elite      # 优势比例
        self.mutprob = mutprob  # 变异率
        self.maxiter = maxiter  # 最大迭代次数
        self.disp = disp        # 是否打印相关信息

    def __elite_select(self, pop, select_num):
        '''精英选择'''
        scores = sorted([(self.fitness(vec, *self.param), vec) for vec in pop])
        ranked = [vec for score, vec in scores]
        return ranked[:select_num]

    def sapga(self):
        '''
        单亲遗传算法主程序，输入的 bounds 应该是一个取值的闭范围，且所有的 bounds
都应该相等
        '''

        topelite = int(self.elite * self.popsize)
```

```
        x_min = self.bounds[0][0]
        x_max = self.bounds[0][1]

        pop = []
        for i in range(self.popsize):
            pop.append(random.sample(range(x_min, x_max + 1), len(self.bounds)))

        for i in range(self.maxiter):
            pop = self.__elite_select(pop, topelite)  # 选出精英个体

            if self.disp:
                print(f"This is NO.{i+1} iteration, the optimal solution is
{pop[0]}, f(x) = {self.fitness(pop[0], *self.param)}")

            # 变异 or 交叉
            while len(pop) < self.popsize and i != self.maxiter - 1:
                variant = random.randint(0, topelite - 1)
                if random.random() < self.mutprob:
                    pop.append(self.__single_parent_mutate(pop[variant].copy()))
# 随机选择一个个体进行变异
                else:

pop.append(self.__single_parent_recombina(pop[variant].copy())) # 随机选择一个个
体进行重组
        print(f"Optimal Solution:{pop[0]}, Fitness Value:{self.fitness(pop[0],
*self.param)}")
        return pop[0]

    def __single_parent_mutate(self, vec):
        '''变异操作，随机修改为一个不重复的值，若为完全指派矩阵，则变异操作等同于
进行重组操作'''
        if len(range(self.bounds[0][0], self.bounds[0][1] + 1)) == len(vec) or
round(random.random()):
            if round(random.random()):
                i = random.randint(1, len(vec) - 1)
                vec = vec[i:] + vec[:i]
            else:
                vec = random.sample(vec,len(vec))
        else:
            i = random.randint(0, len(self.bounds) - 1)
            vec[i] = random.sample(set(range(self.bounds[0][1] + 1)) - set(vec),
1)
        return vec

    def __single_parent_recombina(self, vec):
```

```
        '''交叉操作，任取两个维度作交换，同时有 1/2 概率继续交换'''
        i = random.sample(range(len(vec)), 2)
        vec[i[0]], vec[i[1]] = vec[i[1]], vec[i[0]]
        if round(random.random()):
            vec = self.__single_parent_recombina(vec)
        return vec
```

代码详见：code/ GeneticAlgorithms.py

代码清单 3　重复实验评估代码

```
'''问题 3 仿真求解算法的评价，由于是多次实验，代码运行时间较长'''
from problem3_1 import main as main1
from PlotParam import Plot_parms
import matplotlib.pyplot as plt
import pandas as pd

fenzu1 = [];fenzu2 = [];fenzu3 = []
for i in range(100):
    print(i)
    data1 = pd.DataFrame(columns=['加工 CNC 编号', '上料开始时间', '下料开始时间
'])
    error1 = pd.DataFrame(columns=['故障 CNC 编号', '故障开始时间','故障结束时间
'])
    data2 = pd.DataFrame(columns=['加工 CNC 编号', '上料开始时间', '下料开始时间
'])
    error2 = pd.DataFrame(columns=['故障 CNC 编号', '故障开始时间', '故障结束时间
'])
    data3 = pd.DataFrame(columns=['加工 CNC 编号', '上料开始时间', '下料开始时间
'])
    error3 = pd.DataFrame(columns=['故障 CNC 编号', '故障开始时间', '故障结束时间
'])

    fenzu1.append(main1(20,33,46,560,25,28,31,data1,error1))
    fenzu2.append(main1(23,41,59,580,30,30,35,data2,error2))
    fenzu3.append(main1(18,32,46,545,25,27,32,data3,error3))

plt.figure()
ax = plt.figure(figsize=(5, 2.8)).add_subplot(111)
ax.plot(range(len(fenzu1)), fenzu1, label='分组 1',color='#4f81bd',)
ax.plot(range(len(fenzu2)), fenzu2, label='分组 2',color='#c0504d')
ax.plot(range(len(fenzu3)), fenzu3, label='分组 3',color='#9bbb59')

img = Plot_parms(ax)
img.style_default()
img.update_font(ylabel='加工成料个数')
img.run(name='result_3_1 评价')
```

代码详见：code/Score1.py