摘要

关键字：决策模型

目录

一、 问题重述

1.1 问题背景

一个智能加工系统由 8 台计算机数控机床 (Computer Number Controller, CNC)、1 辆轨道式自动导引车 (Rail Guide Vehicle, RGV)、1 条 RGV 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 是一种无人驾驶、能在固定轨道上自由运行的智能车。它根据指令能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。[**?**] 通常来说，一个工件成品的完成需要若干步，鉴于 RGV 小车在同一时间只能处理同一种任务，而同时最多可以有 8 台数控机床在运行，因此对 RGV 小车进行正确的调度将显著地提高该智能加工系统的加工效率。在本文中，我们将运用不同模型讨论在不同情形下，如何调度小车能够使该系统达到最高效的工作状态。

1.2 问题提出

根据以上的背景，我们在本文中需要针对下面的三种具体情况：

- 一道工序的物料加工作业情况，每台 CNC 安装同样的刀具，物料可以在任一台 CNC 上加工完成；
- 两道工序的物料加工作业情况，每个物料的第一和第二道工序分别由两台不同的 CNC 依次加工完成；
- CNC 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。

完成两项任务：

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：利用已给出的系统作业参数的 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。

二、 模型的假设

基于赛题中给出的一些条件和生活常识，我们在本文中提出如下假设。此后在用到这些假设时，我们将不再另作声明。

- RGV 小车需要对某台数控机床上下料作业时，数控机床对应的传送带上一定有一个预备好的原料工件；
- RGV 小车不能同时进行两项或以上的工作 (包括移动)；
-

三、 符号说明

符号	意义
alg	调度算法/调度方案
n_t	在 t 时间内加工完的工件数
10	421.0
20	640.2

表 1　文中用到的符号和含义

四、 问题分析

我们将针对本问题给出的三种情况分别予以分析，并提出可能的优化算法。

4.1 情况一、二分析

在情况一中，一个工件只需经过一步加工就可以成为成品，然后经过 RGV 小车清洗后放入下料传送带送出该系统。此时，一个工件从原料到成品需要 RGV 小车的如下操作：

1. 移动步骤：RGV 小车从原来的位置运行到 CNC 的位置；
2. 第一次上下料步骤：小车将上料传送带上的原料取下，将 CNC 中的成品置换为原料，若 CNC 原来处于空置状态，则只将原料放入 CNC 中；
3. 等待：等待 CNC 加工完成，此时小车可以进行其他任务；

4. 第二次上下料步骤：小车再次运行到 CNC 的位置，将 CNC 中的成品置换为原料，此时小车不能为已装载成品的状态；
5. 清洗步骤：小车清洗成品，并将成品置入下料传送带上。

而在情况二中，一个工件需要两步加工才能制成成品，因此工件由原料到成品需要多一次上下料步骤，即在第一轮加工完成后，将加工完成的工件放入第二步工序的 CNC 中加工。其他的步骤均是相同的。

显然，因为不存在任何随机因素，因此所有小车调度方案的数量是有限的，故情况一、二的全局最优解是一定存在并且确定的，意即：

$$\exists \text{alg s.t. } n_{t,\text{alg}} = \text{Max}[n_t]$$

所以我们需要找到该最优的调度方案或接近最优的调度方案。

要得到最优的调度方案，可能可以使用的算法有：穷举法、模拟退火算法、决策树算法、遗传算法、神经网络等。其中，穷举法和模拟退火算法的时间复杂度过高，超过了我们所拥有的算力极限，故我们不考虑用此两种方法。同样是基于统计的算法，遗传算法在该题中表现出比模拟算法更优的时间性能，因此遗传算法将成为我们探寻最优解的一种尝试。

4.2 情况三分析

对于情况三，由于加入了随机因素（CNC 发生故障），因此不存在固定的全局最优解。在此情况下，调度过程转化成了小车的决策过程。空闲的小车需要灵活地根据当前加工系统的状态来决定下一步将要进行的步骤。在该情况下，遗传算法仍然可以使用，但是由原来的提供静态方案转变为即时演算出动态决策。

五、 模拟环境

为了能够实际测试调度算法产生的指令序列在 RGV 智能加工系统中的表现，我们基于题目中给出的文档与数据，设计了一个简易的智能加工系统模拟器。每当 RGV 小车处于空闲状态时，模拟器会将当前智能加工系统各部分的状态以及当前的时刻作为参数传递给调度算法，调度算法会以此作为输入进行计算，并且输出当前小车需要执行的指令。

模拟器使用 python 设计，主要由 4 个模块组成。它们分别是 world.py，rgv.py，cnc.py 以及 cargo.py。

world.py 是模拟器的核心部分。在 world.py 中包含模拟器的主对象 World，构造该对象时需要传入工作时间或者加工工件总数，以及调度器作为参数。对象构造完成后，

调用对象的 simulate() 方法就可以开始模拟。模拟结束后，可以调用 result() 和 final() 方法获得加工的工件个数/加工的总时间长，以及每一个工件上料和下料的时间日志记录。

rgv.py 包含了 RGV 对象，其中定义了 RGV 可以接受的指令和它们的编码，它们在模拟器中的对应关系如下表所示。

编码	指令
0	空闲
1	向左移动一格
2	向右移动一格
3	向左移动两格
4	向右移动两格
5	向左移动三格
6	向右移动三格
7	向 #1，#3，#5，#7 号 CNC 上下料
8	向 #2，#4，#6，#8 号 CNC 上下料
9	清洗工件

表 2　RGV 的指令和编码

RGV 对象可以通过 inst() 方法来接受指令。调用这个方法时需要传入一个有效指令的编码，同时根据指令的种类传入一个可选参数，类型为代表工件的 Cargo 类。关于 Cargo 类的信息会在下文描述。调用该方法后会设置 RGV 内部的一些属性，包括对象内部的计时器。

同时，RGV 对象也拥有 update() 方法，调用这个方法时，如果 RGV 正在执行空闲之外的指令，那么程序会将 RGV 内部的计时器减少 1 秒。接着，程序将会检查 RGV 内部的计时器是否已经达到零。如果计时器已经达到零，那么表明当前指令已经执行完成，那么 RGV 对象会根据指令的不同执行一些动作，比如修改 CNC 机床内部的工件对象或者修改自身内部的工件状态。

cnc.py 包含了 CNC 对象。CNC 对象的可能状态如下表所示：

与 RGV 对象类似，CNC 对象也拥有 inst() 方法与 update() 方法。它们的功能也与 RGV 对象中的对应方法类似。

由于智能加工系统的要求，CNC 对象拥有不同模式。可能的模式如下表所示：

编码	指令
0	空闲
1	加工中
2	加工完成
3	故障

表 3　CNC 对象的可能状态

编码	状态
0	一阶段加工
1	二阶段加工，第一阶段
2	二阶段加工，第二阶段

表 4　CNC 对象的可能模式

cargo.py 内部包含 Cargo 对象。Cargo 对象的状态如下表所示：Cargo（工件）对象

编码	状态
0	未加工
1	半加工
2	加工完成，未清洗
3	加工完成，已清洗

表 5　Cargo（工件）对象的可能状态

的状态可以用 RGV 对象和 CNC 对象中的相应指令来修改。同时，保存工件的状态也使得模拟器能够检测出输入的指令序列中的错误，防止不能出现的情况发生。

为了调试方便，我们为模拟器添加了输出函数 info()。输出示例如下：

```
supply cargo 2
```

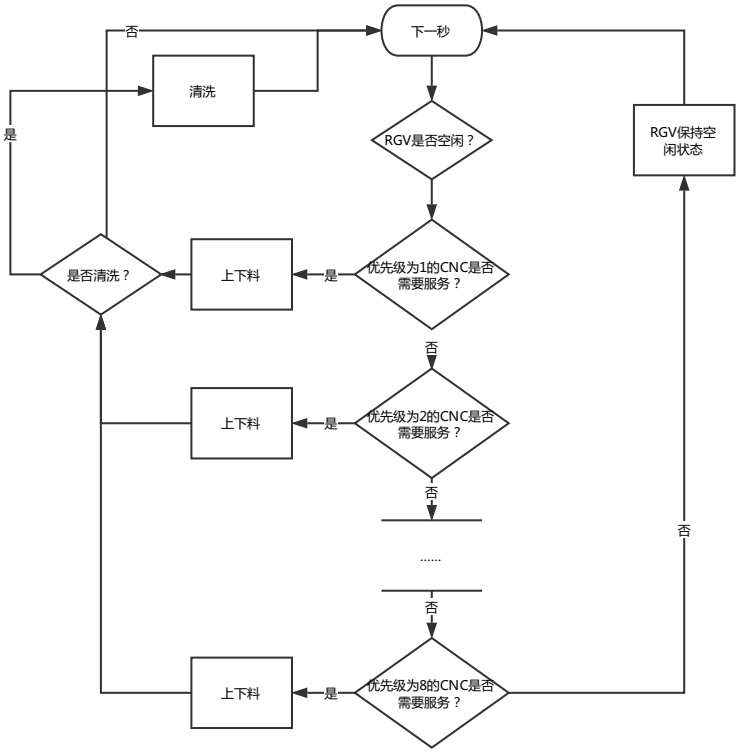

图 1　标准模型（情况一）的运行流程

```
Clock: 3931
RGV:      CNC 1:    CNC 2:    CNC 3:    CNC 4:    CNC 5:    CNC 6:    CNC 7:    CNC 8:
idle      processed processed idle      idle      processed processing idle      idle
8, ready  4, ready  5, ready                      7, ready  9, raw
current cargo: 9
```

模拟结束后，模拟器会将数据写入对应的文件中。

六、 标准调度模型

为了能够定量的评价后续模型的优劣，我们按照题意设计了一个标准调度模型，来与之后的模型对比。对于情况一，标准调度模型基于以下几条原则：

- 小车会优先执行完当前的指令队列，然后再接收 CNC 的服务请求；
- 在所有 CNC 都工作时，若小车的执行队列为空，则小车处于空闲状态；
- 若只有一台 CNC 处于可服务状态，小车将会为这台 CNC 提供服务；
- 若有多台 CNC 处于可服务状态，小车将会优先服务近的、奇数号的 CNC；
- 在完成一次上下料作业后，若需要清洗，小车会立即进行清洗。

七、 遗传算法

7.1 简介

遗传算法在 RGV 调度中有着广泛的应用。Runwei Cheng 等人认为 [**?**] 遗传算法之所以有效，是因为遗传算法能交替在解空间和编码空间之间切换。通过染色体编码将问题转换至编码空间，通过选择、交叉重组和突变等模拟自然遗传的过程寻找最优解，再通过解码将获得的编码转换成解。

7.2 编码方式

染色体的编码决定了计算难度。在车间调度领域，遗传算法的编码方式主要有如下几种：

1. 编码操作
2. 编码任务
3. 偏好队列编码
4. 任务对关系编码
5. 完成时间编码
6. 机器编码
7. 随机密钥编码

在本题中，我们使用的是基于数控机床序号的编码。使用序号编码的好处在于任何编码的顺序都是合法的，编码顺序代表了 RGV 遍历机器的顺序。表 6 给出了一个染色体编码的例子。表中的染色体编码说明小车将按 12345678 的路线为 CNC 上下料。

表 6　染色体编码示例

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

7.3 解码

在 `ga.py` 中，`decode()` 函数负责将染色体编码解释为 RGV 的运动路线和命令。染色体解码遵循以下原则：

1. RGV 会严格地执行染色体编码上的 CNC 遍历顺序。例如，如果 RGV 在编码中指定的 CNC 前就位，但是 CNC 仍在加工工件，RGV 会选择在 CNC 前等待；
2. RGV 会提前在下一台 CNC 前就位。

`decode()` 函数的返回值是一个函数。该函数能够被用于初始化 `World` 对象，控制 RGV 小车移动。

7.4 适应度

`fitness()` 函数计算种群中所有染色体的适应度。每一个染色体的适应度就是染色体对应的RGV运动路线在八小时内能够生产的工件的数量。`select_new_population()` 根据计算得到的适应度选择新一代。所有染色体的适应度使用 Softmax 函数归一，作为该个体在这一代选择中生存下来的概率。使用 Softmax 函数而不是直接用某一染色体的适应度处以所有适应度之和，是因为我们发现在初代中尽管染色体之间相差很大，但由于初代染色体是随机产生的，适应度非常接近。为了将被选择的概率拉开，使用了含有指数的 Softmax。

7.5 重组与突变

模拟重组和突变的函数分别是 `crossover()` 和 `mutation()`。`crossover()` 实现了单染色体交叉点重组。突变会改变染色体上的单个编码。重组与突变的概率均可通过其参数来调整。

7.6 遗传流程

函数 `ga()` 通过调用前文提及的函数对初始种群进行反复迭代，有以下几步。

1. `initial_population()` 产生初代种群；
2. 在循环中先计算每一代的适应度，根据适应度新一代被选择出来；
3. 选择得到的新一代进行重组和突变后，输出其适应度，再进入下一代选择。

7.7 遗传算法成果

以下是遗传算法运行样例（只有十代）：

```
0 [140. 141. 141. 141. 141. 141. 141. 142. 144. 144. 145. 145. 145. 146.
 146. 146. 146. 146. 146. 146. 146. 146. 146. 147. 148. 149. 150.
 150. 152.]
1 [144. 145. 146. 146. 146. 146. 146. 146. 146. 148. 148. 148. 149. 150.
 150. 150. 150. 150. 151. 151. 151. 151. 152. 152. 152. 152. 152. 152.
 152. 154.]
2 [146. 148. 149. 149. 149. 150. 150. 150. 151. 152. 152. 152. 152. 152.
 152. 152. 152. 154. 154. 154. 154. 154. 154. 154. 154. 154. 154. 154.
 158. 158.]
3 [152. 152. 153. 154. 154. 154. 154. 154. 154. 154. 154. 154. 154. 154.
 154. 154. 155. 156. 157. 158. 158. 158. 158. 158. 158. 158. 158. 158.
 158. 159.]
4 [152. 153. 154. 154. 154. 154. 154. 154. 155. 155. 156. 156. 157. 158.
 158. 158. 158. 158. 158. 158. 158. 158. 158. 158. 159. 159.
```

```
 161. 161.]
5 [154. 154. 154. 155. 156. 157. 157. 158. 158. 158. 158. 158. 158. 158.
 158. 159. 159. 159. 159. 159. 160. 160. 160. 161. 161. 161. 161. 161.
 161. 161.]
6 [154. 156. 158. 158. 158. 158. 158. 158. 158. 158. 158. 158. 158. 158.
 159. 159. 160. 160. 161. 161. 161. 161. 161. 161. 161. 161. 161. 163.
 163. 163.]
7 [156. 157. 159. 160. 161. 161. 161. 161. 161. 161. 161. 161. 162. 163.
 163. 163. 163. 163. 163. 163. 163. 163. 163. 163. 163. 163. 163. 165.
 166. 166.]
8 [158. 161. 161. 161. 161. 162. 163. 163. 163. 163. 163. 163. 163. 163.
 163. 163. 163. 163. 165. 165. 166. 166. 166. 166. 168. 168. 168. 168.
 168. 172.]
9 [163. 163. 163. 165. 167. 168. 168. 168. 168. 168. 168. 168. 168. 168.
 168. 168. 169. 169. 169. 171. 171. 171. 171. 171. 172. 172. 172. 172.
 172. 172.]
```

遗传算法得到的结果见表 **??**。

表 7　遗传算法结果

迭代代数	八小时内加工工件个数
10	172
100	204
500	238

table 环境是一个将表格嵌入文本的浮动环境。tabular 环境的必选参数由每列对应一个格式字符所组成：c 表示居中，l 表示左对齐，r 表示右对齐，其总个数应与表的列数相同。此外，`@{文本}` 可以出现在任意两个上述的列格式之间，其中的文本将被插入每一行的同一位置。表格的各行以`\\`分隔，同一行的各列则以 & 分隔。`\toprule`、`\midrule` 和`\bottomrule` 三个命令是由 booktabs 宏包提供的，其中`\toprule` 和`\bottomrule` 分别用来绘制表格的第一条（表格最顶部）和第三条（表格最底部）水平线，`\midrule` 用来绘制第二条（表头之下）水平线，且第一条和第三条水平线的线宽为 1.5pt，第二条水平线的线宽为 1pt。引用方法：“如表 `\ref{标签名}` 所示”。

参考文献

[1] 2018 年全国大学生数学建模竞赛 B 题，cumcm.cnki.net，2018.9.13

[2] Cheng, Runwei, Mitsuo Gen, and Yasuhiro Tsujimura. ”A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation.” Computers & industrial engineering 30.4 (1996): 983-997.

附录 A 排队算法–matlab 源程序

```
kk=2;[mdd,ndd]=size(dd);
while ~isempty(V)
[tmpd,j]=min(W(i,V));tmpj=V(j);
for k=2:ndd
[tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
tmp2=V(jj);tt(k-1,:)=[tmp1,tmp2,jj];
end
tmp=[tmpd,tmpj,j;tt];[tmp3,tmp4]=min(tmp(:,1));
if tmp3==tmpd, ss(1:2,kk)=[i;tmp(tmp4,2)];
else,tmp5=find(ss(:,tmp4)~=0);tmp6=length(tmp5);
if dd(2,tmp4)==ss(tmp6,tmp4)
ss(1:tmp6+1,kk)=[ss(tmp5,tmp4);tmp(tmp4,2)];
else, ss(1:3,kk)=[i;dd(2,tmp4);tmp(tmp4,2)];
end;end
dd=[dd,[tmp3;tmp(tmp4,2)]];V(tmp(tmp4,3))=[];
[mdd,ndd]=size(dd);kk=kk+1;
end; S=ss; D=dd(1,:);
```

```
kk=2;
[mdd,ndd]=size(dd);
while ~isempty(V)
   [tmpd,j]=min(W(i,V));tmpj=V(j);
for k=2:ndd
   [tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
   tmp2=V(jj);tt(k-1,:)=[tmp1,tmp2,jj];
end
   tmp=[tmpd,tmpj,j;tt];[tmp3,tmp4]=min(tmp(:,1));
if tmp3==tmpd, ss(1:2,kk)=[i;tmp(tmp4,2)];
else,tmp5=find(ss(:,tmp4)~=0);tmp6=length(tmp5);
if dd(2,tmp4)==ss(tmp6,tmp4)
   ss(1:tmp6+1,kk)=[ss(tmp5,tmp4);tmp(tmp4,2)];
else, ss(1:3,kk)=[i;dd(2,tmp4);tmp(tmp4,2)];
end;
end
   dd=[dd,[tmp3;tmp(tmp4,2)]];V(tmp(tmp4,3))=[];
   [mdd,ndd]=size(dd);
   kk=kk+1;
end;
S=ss;
D=dd(1,:);
```