**智能 RGV 的动态调度策略**

摘要

研究智能 RGV 的动态调度对提高智能加工系统工作效率具有重要意义，本文针对该

问题，采用基于排队论的高响应比优先调度算法，利用计算机仿真模拟智能系统工作过

程，建立 RGV 的动态调度模型并利用改进后的遗传优化算法对仿真调度模型进行优化。

针对任务 1，以系统作业效率为目标函数，而系统作业效率的直接衡量指标是 8 小

时内系统实际完成加工物料的数量，为使目标函数取得最大值，约束条件为总作业时间

限制、等待状态参量限制、故障状态参量限制、操作原则与被操作对象限制、加工物料

完成标准限制、CNC 及 RGV 状态参量限制、CNC 刀具类型限制。据此建立计算机仿真

模型，当系统只有一种刀具时，各 CNC 类型指示有唯一取值；当系统存在两种及两种以

上刀具时，则需要将各道流程工序的刀具分配安装到不同 CNC 上，会产生多种组合，当

规模较大时，程序复杂度指数增长求解不易，故考虑采用遗传算法模型，结合本题背景，

对遗传算法做出改进。将改进的遗传优化模型和计算机仿真模型相结合得出动态调度模

型。

针对任务 2，关于 RGV 的调度策略，对于只有一道工序的物料加工，即 Case\_1，

Case\_3\_1，由于刀具分配情况只有一种，其调度策略为直接使用以优先级排序调度为核

心的计算机仿真调度；对于有两道工序的物料加工，即 Case\_2，Case\_3\_2，由于刀具分

配情况有多种，故其调度策略为：先使用改进的遗传算法，对刀具分配组合进行优化，

再利用计算机仿真调度，得到针对需要两道加工工序的物料加工时刀具分配的较好调度

方案。考虑以 8 小时内系统加工完成的物料数来衡量不同系统的作业效率，采用各自的

调度策略，分别对三组智能加工系统进行计算机仿真模拟，得到三组系统在四种情况下

的作业效率，第一组系统在 Case\_1、Case\_2、Case\_3\_1、Case\_3\_2 情况下的作业效率分

别为 359、305、350、288；第二组系统在四种情况下的作业效率（单位：个/8h）为 338、

265、323、196；第三组系统在四种情况下的作业效率为 368、333、358、269。在实际

的生产过程中，CNC 的数量和刀具的类型较多，本文所建立的模型充分考虑到这些问

题，分别采用在*n* 个 CNC，*m* 种刀具情况下的普适处理和在 CNC 数量、刀具种类规模

较大情况下的组合优化方法；并且该模型不仅仅对于题目中所给的一般的直行传送带有

效，对于任意的联通的通道也适用，故该模型对于一般问题具有较好的适用性。关于算

法有效性考虑分别从算法收敛性、算法求解结果的正确性、算法复杂度、算法求解精确

度等方面分析得出本文设计算法可行性较高的结论，这与代码运行结果的检验是一致的。

本文已建立的计算机仿真模型只适合一个 RGV 的情况，但在真正的作业车间里，

往往作业量较大，一个 RGV 可能不足以应对，所以考虑多个 RGV 的情况，将 RGV 的

固定轨道拓展为任意联通道路，将拓展轨道的转角视为结点，从而将整个拓展轨道看作

一个无向带权连通图*G* 。考虑到实际模拟情况下，按照最小时间分度进行检测，可能会

增大系统计算量，导致系统开销过大，因此可以稍微减少时间精度，以获得较大的系统

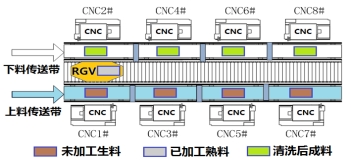
开销减少。

关键词：高响应比优先调度 排队论 计算机仿真 动态调度模型 改进遗传优化算法

1

|  |  |
| --- | --- |
| *N* | 加工物料总数 |
| *hour* | 加工时长 |
| *n* | CNC 总量 |
| *m* | 加工流程 |
| *P* | 优先级矩阵 |
| *M* | 加工物料记录表 |

一、 问题重述



1.1 问题的背景

图 1 是一个智能加工系统的示意图，由 8 台计算机数控机床（Computer Number

Controller，CNC）、1 辆轨道式自动引导车（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。

**图 1：智能加工系统示意图**

针对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 安装同样的刀具，物料可以在任一

台 CNC 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 依次加工完成；

（3）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。

1.2 问题的提出

问题一：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；

问题二：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、 模型假设

2.1 假设 RGV 执行命令全过程不会出现故障。

2.2 不考虑 RGV 接收、处理信号的时间。

2.3 假设人工维修 CNC 的时间服从正态分布。

2

三、 符号说明

|  |  |
| --- | --- |
| *mi*1 | 加工物料序号 |
| *B*6*n* | CNC 状态参量矩阵 |
| *b*1*j* | 指示第*j* 个 CNC 的工作状态 |
| *b*2*j* | 指示第*j* 个 CNC 加工物料类型，即负责加工第*k* 道工序 |
| *b*3*j* | 指示第*j* 个 CNC 上存在的物料编号 |
| *b*4*j* | 指示第*j* 个 CNC 本个物料加工已工作时间 |
| *b*5*j* | 第*j* 个 CNC 上个物料加工完成时间 |
| *b*6*j* | 第*j* 个 CNC 故障状态 |
| *bR* | RGV 状态参量 |
| *lR* | RGV 位置参量 |
| *MtRj* | RGV 移动到第*j* 个 CNC 所需要时间 |
| *It j* | 第*j* 个 CNC 上下料所需要的时间 |
| *tw* | RGV 完成一个清洗作业所需时间 |
| *Ct j* | 第*j* 个 CNC 加工第*r* 道工序所需要的时间 |
| *Ts j* | 当前时刻、位置，RGV 移动到完成操作第*j* 个 CNC 所需时间 |
| *Wt j* | 第*j* 个 CNC 已等待的时间 |



四、 动态调度模型的建立与求解



4.1 模型的分析与建立

4.1.1 模型的准备

设定工作时间段内 CNC 成功加工完成物料的总数为*N* ，加工时长为*hour* ，一定时

间内完成加工的物料个数可以反映该系统的作业效率。设定 CNC 总数量为*n* ，加工流程

（即一个物料完成加工需要的工序）为*m* ，其中*m* *n* ；CNC 状态参量矩阵为*B*6*n* ，其

中*b*1*j* (1 *j* *n*) 指示第*j* 个 CNC 的工作状态，

3

1

*b*1*j* = 

0

CNC#*j*处于工作状态

CNC#*j*处于空闲状态

*b*2*j* (1 *j* *n*) 指示第*j* 个 CNC 加工物料类型，即负责加工第*k* 道工序，

*b*2*j* =*r* (1 *r* *m*)

*b*3*j* (1 *j* *n*) 指示第*j* 个 CNC 上放置的物料的编号，

*b*3*j* =*k* (*k*  1)

*b*4*j* (1 *j* *n*) 指示第*j* 个 CNC 本个物料加工已工作时间；

*b*5*j* (1 *j* *n*) 指示第*j* 个 CNC 上个物料加工完成时间；

*b*6*j* (1 *j* *n*) 指示第*j* 个 CNC 故障状态，

1

*b*6*j* = 

0

CNC#*j*处于正常状态

CNC#*j*处于故障状态

RGV 的状态参量为*bR* ，

1

*b*1*j* = 

0

RGV处于工作状态

RGV处于空闲状态

 *n*  

*lRj* *R* −    

RGV 的位置参量为*lR* ，其中 1 *lR*     ；

  2  

RGV 与第*j* 个 CNC 之间的距离为*lRj* ，

*j* 

=*l*

 2 

RGV 移动到第*j* 个 CNC 所需要时间为*MtRj* ，

*MtRj* =*ti* ，*lRj* =*i*

RGV 为第*j* 个 CNC 上下料所需要的时间为*It j* ，

*t*奇 ，*j* mod 2 = 1

*It j* = 

*t*偶 ，*j* mod 2 = 0

RGV 完成一个物料的清洗作业所需的时间为*tw* ；

4

第*j* 个 CNC 加工第*r* 道工序所需要的时间为*Ct j* ，



*Ct j* =*tr* (1 *r* *m*)

在当前时刻、位置，RGV 关于第*j* 个 CNC 完成操作所需要的时间一共为*Ts j* ，

*b* 

*Ts j* =*Mt j* +*It j* +  2*j* *tw*

*m* 

RGV 原地等待一次的时间为*Tw* ，第*j* 个 CNC 目前已等待的时间为*Wt j* 。

设关于 CNC 的排序优先级为*p* ，

*p j* =

*Ts j* +*Wt j*

*Ts j*

记待处理 CNC 列表为*W* ，其中*wi*1,*wi*2 分别表示待处理 CNC 的序号以及该 CNC 当前时

刻等待时间。对优先级矩阵*P* ，加工物料记录表为*M* ，*mi*1 表示加工物料序号，其中*mi*,3*r*−1

表示操作第*r* 道工序加工的 CNC 编号，*mi*,3*r* 表示上料开始时间，*mi*,3*r*+1 表示下料开始时

间；设定第*r* 道工序的所需原材料集合为*Qr* ，其中*Q*1 中元素的个数为无穷个；记 RGV

的 第*x* 次 操 作 为*Opx* ， 操 作 对 象 记 为*Opx*.*C* ； 故 障 处 理 记 录 表 为*ACC* ， 其 中

*acci*1,*acci*2,*acci*3,*acci*4 分别表示故障序号，故障 CNC 的编号，故障开始的时间，故障 CNC

被修好的时间。

4.1.2 模型的建立

（1）、计算机仿真模型

实际系统作业时，往往追求系统效率最高，该问题中，效率的直接衡量指标是一定

时间段内系统实际完成加工物料数量的多少，故目标函数为

max*N* = 

*m*3*m*+1

*m*3*m*+1

1).总作业时间限制

 (*Ts* +*Tw*) *hour*

2).等待状态参量限制

(

)

*b*1*wi*1*b*6*wi*1 = 1



3). 故障状态参量限制

5

*b*1*acci*2 +*b*6*acci*2 = 0



4). 操作原则与被操作对象限制

*Opx*.*C* = arg max*p* (*j* ) ,*x*  1

(

*b*1*j* )*b*6*j* = 1



*Qb*2*j*  

5). 加工物料完成标准限制

*hour* −*m*3*m*+1 *Ctm*3*m*−1

6). CNC 及 RGV 状态参量限制

*b*1*j* *b*6*j*

*bR* + *b*1*jb*6*j*  1

*n*

*j*=1

*b*1*j* ,*b*6*j* ,*bR* 0,1

7). CNC 刀具类型限制

*b*2*j* 1, 2,3,

,*m*

则目标函数及约束条件为：

max*N*

*N* = *m*

(

)

*Opx*.*C* = arg max*p* (*j* ) ,*x*  1

(*b*1*j* )*b*6*j* = 1

 (*Ts* +*Tw*) *hour*

*b*  ,*b*  ,*b*  0,1

 *m*3*m*+1

 3*m*+1

*hour* −*m*3*m*+1 *Ctm*3*m*−1



 *b*1*wi*1*b*6*wi*1 = 1



*b*1*acci*2 +*b*6*acci*2 = 0



 



*Qb*2*j*  



 *n*

*bR* + *b*1*jb*6*j*  1

 *j*=1



*b*1*j* *b*6*j*

 1*j* 6*j* *R*

*b*2*j* 1, 2,3, ,*m*

6

（2）、改进遗传优化模型



当*m* = 1，上述（1）中的计算机仿真模型中的 CNC 类型指示有唯一取值，*b*2*j* = 1(1 *j* *m*) ；

当*m*  2 时，上述（1）中的计算机仿真模型中的 CNC 类型指示取值不唯一，其中

*b*2*j* 1, 2,3,

,*m*(1 *j* *n*) ，那么则存在一种或多种组合，使得目标函数取值*N* 最大，

将其中每一种组合记为*V* ，

*V* = (*v*1,*v*2 ,

,*vi* ,

,*vn* ) ,1 *vi* *m*

根据分析，容易得到，对于有*m* 道流程的物料，缺少其中任意一道流程都是不可行的，

记*S* (*m*) 为将*m* 种刀具分配安装到*n* 个 CNC 上的可行组合，根据排列组合知识可以得知

*m*−1

*S* (*m*) =*mn* − 

*i*=1

*m*!

(*m* −*i*)!

*S* (*m* −*i* )

结合算法知识，可以得知，进行遍历组合，得出的程序复杂度将会在*O* (*nn* ) ，复杂度呈

现指数增长，对于小规模问题，可以求解，但对于大规模问题，求解不易，故考虑采用

遗传算法模型，结合题目背景，对遗传算法做出改进。

1). 编码方式

每一种组合*V* 可以看作一次编码，编码*v*1*vv*2 3

*vn* 作为染色体，每一组染色体对应一

个个体。

2). 初始种群

采用随机数生成的方式，生成多组染色体，留下符合条件（所有流程都包含）的染

色体，留下染色体的数目等于种群大小。

3). 目标函数

采用计算机仿真模型计算系统效率*N* ，将*N* 的值作为目标函数值。

4). 配对原则

对于留下的染色体，根据*N* 的大小，采用大对大，小对小的方式，进行染色体配对，

准备下一步的交叉操作。

5). 交叉操作

采用单点交叉法，统计环境下随机交换一半的基因（每对基因交换的概率是

*pC* = 0.5 ），生成各自子代*V*1,*V*2 。

6). 变异操作

在恒定变异率的情况下，每一个个体的每一个基因可能发生变异，基因突变是不定

向的，所以可以突变为1 ~*m* 中任意一种基因。

结合题目实际情况，自然状态下发生基因突变的个体是任意的，相同个体间突变率

可以看作是无差异的，但这对题目的会产生较大的影响，因为初始染色体是随机的，基

因组合和目标函数值并没有显著的线性关系，所以个体的适应情况（目标函数值的大小）

不一定很可观，过多的不定向变异可能会导致优良个体消失，所以在此对适应度排名靠

前的个体不实行变异。

7). 环境适应度

7

环境适应度满足下面两条原则：

（1）环境适应度用目标函数值表示，目标函数值越大，适应度越强。

（2）当染色体中缺少任何一种基因时，则完全不能适应，直接淘汰。

8). 选择原则

选择原则满足以下两点：

（1）根据*N* 的大小进行排序，保持种群数目不变，其余淘汰。

（2）个体出生或者变异时，检测其环境适应度，完全不能存活的，直接淘汰。

这里考虑将种群大小设为*Z* =100 ，最大代数为*G*=1000 ，交换率*pc* = 1，为保持充分进化，

变异率为*pm* = 0.25 ，由于初始值过于随机，可认定环境不稳定，个体受影响较大，故设

置较大变异率。

4.2 模型的求解算法

4.2.1 调度模型的仿真算法

Step1：初始化参量，执行 Step2。

Step2：检测是否存在有故障的 CNC，如果有，执行 Step3,否则，执行 Step4。

Step3：检测故障的 CNC 是否已修好，更新已修好的 CNC 状态参量，执行 Step4。

Step4：检测是否存在正在工作且无故障的 CNC（不包含刚刚操作过的 CNC），如果存

在，执行 Step5，否则，执行 Step6。

Step5：检测正在工作而且无故障的 CNC（不包含刚刚操作过的 CNC）是否完成工作，

将完成工作的 CNC 按照各自种类，更新状态参量，加入待处理列表，更新待处理列表，

执行 Step6。

Step6：检测是否到达预定时间，如果是，执行 Step16，否则，执行 Step7。

Stpe7：检测待处理列表是否非空，如果是，执行 Step8，否则，执行 Step13。

Step8：根据排序规则，按照 CNC 各自的类型，计算 CNC 的优先级，执行 Step9

Step9：检测优先级最高的是否是第 m 道工序 CNC，如果是，执行 Step10，否则执行

Step14。

Step10：检测当前是否有 m-1 道工序 CNC 生产的半成品，如果有，执行 Step11，否则，

执行 Step14。

Step11：更新 RGV 状态参数，更新 CNC 状态参数，更新物料记录表，将其从待处理列

表中删去，更新待处理列表，模拟该 CNC 是否将会出现故障，如果是，执行 Step12，

否则，执行 Step3。

Step12：模拟故障发生时间，人工维修耗时，加入故障记录表，更新该 CNC 状态参量，

执行 Step3。

Step13：检测当前正在工作的无故障 CNC，如果此种 CNC 存在，选取正在工作的无故

障 CNC 中距离完成时间最近的 CNC 的完成时刻，更新时间 T 到这一时刻，执行 Step3，

如果不存在，执行 Step15

Step14：m 的值更新为 m-1，检测 m 的值是否为 0，如果是，执行 Step13，否则，执行

Step10。

Step15：检测当前正在维修的 CNC，选取正在维修的 CNC 中距离维修完成时间最近的

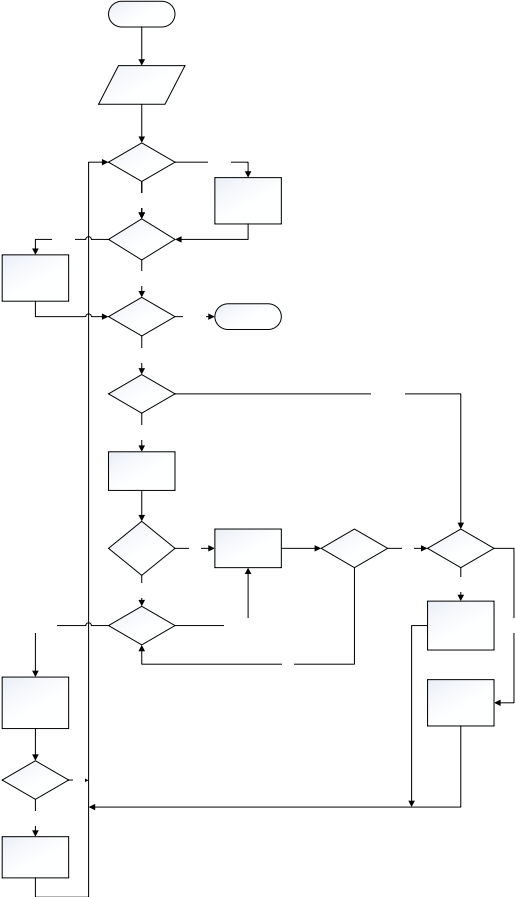
CNC 的维修完成时刻，更新时间 T 到这一时刻，执行 Step3。

Step16：计算加工完成的物料数量，退出程序。

动态调度模型的计算机仿真算法如图 2 所示。

8

开始



初始化

参量

存在故障

CNC？

不存在

存在

更新已修好

的CNC状态

参量

存在

更新工作中

的CNC状态

参量

存在工作中

的CNC？

不存在

到达预定时

间？

未到达

到达

结束

存在空闲的

CNC？

存在

计算优先度

不存在

优先级最高的

CNC 负责第m

道工序？

否

m=m-1

m=0？

是

存在正在工

作的CNC？

是

是

存在

存在m-1道工

序半成品？

不存在

更新T到有

CNC工作完

成的时刻

否

否

RGV操作CNC，

更新RGV、CNC

状态参数，各

项记录表

更新T到有

CNC维修完

成的时刻

该CNC出现

故障？

是

更新CNC状态

参数、故障记

录表

否

图 2：仿真算法流程图

9

|  |  |  |
| --- | --- | --- |
| 情况类型 | 组别 | CNC 刀具分配策略 |
| Case\_2 | 第一组 | （2,1,1,1,2,1,2,1） |
| 第二组 | （2,1,1,1,1,2,1,2） |
| 第三组 | （1,1,1,1,2,2,1,1） |
| Case\_3\_2 | 第一组 | （1,1,1,1,2,2,2,2） |
| 第二组 | （2,1,1,1,2,1,1,2） |
| 第三组 | （1,1,1,1,1,2,2,2） |

4.2.2 调度模型的优化遗传算法



Step1：随机生成第一代种群，计算适应度，并按照适应度对种群中所有个体进行排序，

执行 Step2。

Step2：按照个体适应度顺序，对个体进行交叉操作（例：*Vi* 与*Vi*+1 配对，*i* 为奇数），将

得到的新个体按照选择原则（2）进行淘汰，对当前的种群进行适应度排序，执行 Step3。

Step3：按照变异规则执行变异操作，将变异后的个体按照选择原则（2）进行淘汰，对

当前种群进行排序，执行 Step4。

Step4：按照选择原则（1）进行淘汰，将剩余个体进行排序，检测是否到达设定代数，

是则退出程序，否则进行下一代繁殖，执行 Step2。

改进后的遗传优化算法流程图如图 3 所示。

开始

结束

生成第一代种

群，并按适应

度排序

进行交叉操

作，并按选择

原则（2）进行

淘汰

否

按照变异规

则，进行变异

操作，并按选

择原则（2）进

行淘汰

按照选择原则

（1)进行淘汰,

并对种群进行

适应度排序

达到设定代

数？

是

图 3：改进后的遗传优化算法流程图

五、 RGV 的调度策略和系统作业效率

5.1 RGV 的调度策略

关于 RGV 的调度策略，考虑按照物料加工所需工序分为两类。对于只有一道加工

工序的物料加工，即 Case\_1，Case\_3\_1 两种情况，由于刀具分配情况只能有一种，其调

度策略为直接使用以优先级排序调度为核心的计算机仿真调度模型。而对于有两道加工

工序的物料加工，即 Case\_2，Case\_3\_2 两种情况，由于刀具分配情况有多种，故其调度

策略为：先使用改进的遗传算法，对各台 CNC 对两种刀具的分配组合进行优化，再利用

以优先级排序调度为核心的计算机仿真调度模型，采用改进的遗传算法，利用 Matlab 编

程求解，得出两道加工工序的物料加工情况下，CNC 刀具分配的较好的结果如表 1 所示。

5.2 系统作业效率

表 1：CNC 刀具分配策略汇总表

按照 5.1 中设定好的各种情况下三组智能加工系统的调度策略，利用 Matlab 编程

实现对三组智能加工系统的计算机仿真模拟，得到三组智能加工系统在四种情况下的作

业效率（用 8 小时内生产的成料个数*N* 衡量）如表 2 所示。

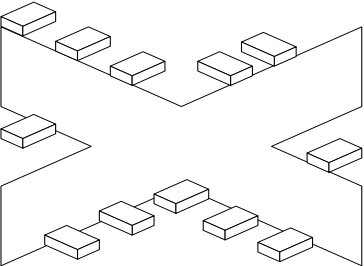
10

|  |  |  |  |
| --- | --- | --- | --- |
| *N* （个/8h） | 第一组 | 第二组 | 第三组 |
| Case\_1 | 359 | 338 | 368 |
| Case\_2 | 305 | 265 | 333 |
| Case\_3\_1 | 350 | 323 | 358 |
| Case\_3\_2 | 288 | 196 | 269 |

加入了机器存在一定故障率的设定，而 Case\_2、Case\_3\_2，则考虑了刀具分配包含多种

表 2：三组数据在所给情况下作业效率汇总表

对于表 2 所列出的三组智能加工系统的作业效率，其中 Case\_3\_1、Case\_3\_2 两种情况



组合的情况。由于使用改进遗传算法有一定可能性使得求得的结果并不是最优解，随机

情况有着一定的不确定性，所以表 2 列出的系统作业效率并不能代表其整体情况，更多

的系统效率检测，可以使用附件中程序进行操作得出。

六、 模型的实用性及算法的可行性检验

6.1 模型的实用性

对于本题采用的计算机仿真模型和改进遗传算法模型，其中计算机仿真模型，是在

充分分析题目实际情况后，在多个 CNC，多种刀具，并且机器存在一定的故障率的情况

下，探求其系统运作内在机理而得出的最大程度符合实际情况的数学模型。而改进遗传

算法模型，则是在原始经典模型的基础上，结合了本题实际情况，得到的符合本题的个

性化改进遗传算法模型。在对三组智能加工系统的四种不同情况下的仿真模拟和模型求

解中，从物料记录表中，通过分析加工物料的 CNC，物料上机和下机时间，可以看出模

型较好地反映了这一物料生产过程。

在实际的生产过程中，CNC 的数量和刀具的类型较多，本文所建立的模型充分考虑

到这种特点，分别采用了*n* 个 CNC，*m* 种刀具情况下的普适处理和 CNC 数量、刀具种

类规模较大情况下的组合优化方法，故该模型对于一般问题具有较好的适用性。

事实上，该模型不仅仅对于题目中所给的一般的直行传送带有效，对于任意的联通

的通道也是适用的，比如图 4 所示的十字型 CNC 分布图，将 RGV 运行道路简化为直

线，CNC 分布于线两侧，那么对于任意一个无向带权连通图*G*（各边权值为各边长度），

本题模型在此种道路下也是适用的，对原模型所做的唯一改变是，将计算 RGV 到每个

CNC 的步数改为计算 RGV 到每个 CNC 的最小运行时间。

图 4：十字型 CNC 分布图

11

|  |  |  |
| --- | --- | --- |
| 情况类型 | 系统效率均值 | 系统效率均方差 |
| Case\_1 | 359 | 0 |
| Case\_3\_1 | 351.1 | 6.227 |

6.2 算法的可行性

1). 算法收敛性

如图 2、图 3 所示，计算机仿真算法和改进遗传算法的算法流程图均有唯一的出口，

所以两个算法都是收敛的。

2). 算法求解结果的正确性

对于题中给出的三组智能加工系统的四种情况，计算机仿真算法按照系统内部机理，

合理地求解出了各组系统在不同情况下的作业效率，以 Case\_1 为例，第三组系统作业

效率高于第一组，而第一组系统作业效率高于第二组，再观察给出的第三组、第一组、

第二组系统的各项参数，发现三组系统的系统作业效率的相对关系与其中 CNC 加工完

成一道工序所需要的时间相对关系是一致的，从直观上看，该仿真算法正确求解了调度

模型。对于改进的遗传算法，在迭代 10 代内，对于题中所给出的四种情况，求出了相

对稳定的较优解，对于这些解，可以再挑选系统效率距离最高系统效率较近的一些点，

排除可能是偶然出现的或固有的偏大的点，从而更加有效地满足平均的系统效率较大化。

3). 算法复杂度分析

本文所设计的计算机仿真算法复杂度为*O* (*n*2 ) ，在次数较低的多项式的情况下，短

时间内可以求解，故其收敛速度较快，经测试，这与程序的实际运行耗时结果一致。

本文所设计的改进遗传算法复杂度为*O* (*n*3 ) ，相对计算机仿真算法，改进遗传算法

需要消耗大量时间（取决于迭代次数），迭代次数较大时，其收敛速度较慢，经测试，这

与程序实际运行耗时结果基本一致。

4). 算法求解精确度分析

计算机仿真算法是在对实际问题剖析透彻，了解机理的基础上建立起来的，因而与

参数的选取有很大的关系，若参数选定合适，并且给定了一个合理的模拟随机量的方式，

那么算法的求解精确度将较高。以第一组系统的 Case\_1 与 Case\_3\_两种情况为例，分别

对两种情况各进行 10 次模拟，得到其系统效率均值和均方差如表 3 所示。

七、 模型的优缺点及改进

表 3：第一组系统的系统效率均值和均方差结果表

观察表 3 可以发现系统的结果误差并非偶然，其精确度是可以接受的。

7.1 模型的优点

本文建立的计算机仿真模型具有较强的适用性，不仅满足题目中所列出的四种情况，

同时对在有多个 CNC（大于 8 个），多种刀具（大于 2 个），并且机器存在一定的故障率

（故障率可调整）的情况下也适用。建立模型时进行了采用*n* 个 CNC，*m* 种刀具情况下

的普适处理和 CNC 数量、刀具种类规模较大情况下的组合优化问题处理；并且该仿真

模型不仅仅局限于题目中所画的水平流水线型传送带，而是对于任意一个无向带权连通

图*G* （各边权值为各边长度），该模型也是适用的，而相对于现模型所做的唯一调整就

是，将计算 RGV 到每个 CNC 的步数改为计算 RGV 到每个 CNC 的最小运行时间，因

而该模型对于一般问题具有较好的适用性。针对该模型所设计的仿真算法在时间上具有

较好的收敛性，并且经过测试可以发现模型求解结果基本稳定。本文针对模型所设计的

12

代码已趋于成熟，读者可自行运行参考验证结果。



7.2 模型的缺点

对于计算机仿真模型，本文仅考虑了 1 个 RGV 的情况，未推广到*n* 个 RGV 的情

况。对于针对调度模型所设计的改进遗传算法，其具有较高的时间复杂度，当迭代次数

较大时（大于 10 代）在时间上的收敛性较差，由于时间原因，无法大量进行超过百代

的迭代试验，从而会有一定可能出现不是最优解的情况。

7.3 模型的改进

本文已建立的计算机仿真模型只针对 1 个 RGV 的情况，但在真正的作业车间里，

作业量较大，一个 RGV 可能不足以应对，所以考虑多个 RGV 的情况，按照模型实用性

里的关于 RGV 工作轨道的分析，将 RGV 的固定轨道拓展为任意连通道路，将拓展轨道

的转角视为结点，从而将整个拓展轨道看作一个无向带权连通图*G* ，如图 5 所示，拓展

轨道两侧分布有 CNC。

*g*1

*RGV* #1

*g* 6

*g*9

*g*2

*g*4

*RGV* #3

*g*5

*g*8

*g*3

*RGV* # 2

*g*7

图 5：拓展轨道连通图

在新的拓展轨道下的多 RGV 情况，对每一个 RGV 而言，计算其与场上每一个 CNC 的

可达（可以找到一条路 RGV 前去工作）状态，对于每一个可达的 CNC，RGV 将计算其

到该 CNC 的最短时间，一般来说，最短时间与行走的路径有关，记 RGV 到 CNC#*j* 的

某条路径为：

*pathx* = (*gx*1 ,*gx*2 ,

,*gxend*

)

现在考虑步长问题设步长为：

*lRj* = (*gx*1 −*lbegin* ) + (*lend* −*gxend*

)  (*g*

*end* −1

*i*=1

*xi*+1

−*gxi*

)

再考虑转角问题，一般来说，机器在弯道转向，需要更长的时间，设

*f*1 (*lRj* )

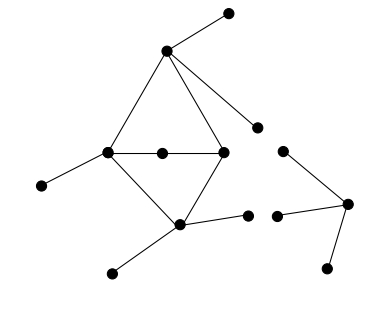
*Mt* = 

*f*2 (*end* )

其中*f*1,*f*2 分别是步长数和转弯数与耗费时间的映射关系。

13

那么其在路径*x* 下到达目标 CNC，所用总的时间：



*Mtxj* =*f*1 (*lRj* ) +*f*2 (*end* )

取到该 CNC 所有路径中耗时最短的路径作为 RGV 到该 CNC 的目标路径。原来的优先

级计算方法不变，若同一 RGV 对多个 CNC 优先级相同，那么可以选择：

（1）距离步数最短的 CNC。

（2）距离弯道最少的 CNC。

选取哪一种可以根据仿真测试结果进行调整。若一个 CNC 对多个 RGV 优先级最高，那

么也有两种选择：

（1）距离步数最短的 RGV。

（2）距离弯道最少的 RGV，

选取哪一种可以根据仿真测试结果进行调整。

由于有多个 RGV 协同工作，RGV 之间有相互通信，将原算法改为按照最小时间分度进

行检测，在每一时刻，每一 RGV 检测可达 CNC 时，将场上其他的 RGV 看作断点，如

图 6 所示。

*g*1

*g* 6

*g*9

*g*2

*RGV* #1

*g*3

*g*11

*g*4

*g*12

*g*10

*g*13

*g*5

*g*8

*g*7

图 6：单个 RGV 视角下拓展轨道连通图

每前进一个时刻，所有 RGV 各自进行计算，并信息共享，再次按照上述规则进行执行。

考虑到实际模拟情况下，按照最小时间分度进行检测，可能会增大系统计算量，导致系

统开销过大，因此可以稍微减少时间精度，以获得较大的系统开销减少。

八、 参考文献

[1].姜启源、谢金星、叶俊. 数学模型. 北京：高等教育出版社，2011 年。

[2].司守奎、孙兆亮、数学建模算法与应用. 北京：国防工业出版社，2017 年。

[3].吴焱明、刘永强、张栋、赵韩，基于遗传算法的 RGV 动态调度研究，《起重运输机

械》，（2012）06-0020-04，2012 年。

[4].谢胜利、董金祥、黄强，基于遗传算法的车间作业的调度问题求解，《计算机工程与

应用》，（2002）10-0079-04：2002 年。

[5]Norharyati Md Ariff, Noor Elaiza Abdul Khalid, Rathiah Hashim, Noorhayati Mohamed

14

Noor. Selfish Gene Algorithm Vs Genetic Algorithm: A Review[J].IOPConference Series:

Materials Science and Engineering,2016,160(1).

[6].Tremaine, Robert L..The Promise of Computer Simulation Applications.[J].Defense & AT-

L,2009,38(1):51-55.

[7].Areeg Abdalla and James J. Buckley.Monte Carlo methods in fuzzy queuing theory[J].Soft

Computing - A Fusion of Foundations, Methodologies and Applications,2009,13(11)1027-1033.

[8].D.R.W. Holton, M. Younas, l.U. Awan.Priority scheduling of requests to web

portals[J].Journal of Systems and Software,2011,84(8):1373-1378.

15

附录

1、仿真模拟程序

clear

n=8;%设定 CNC 数目

b=zeros(5,n);%初始化 CNC 状态指示（1）&物料指示（3）&时间指示（4）&停机指示

（5）

%%%%%%%%参数修改处

r=0; m=1; move=[0,20,33,46]; CNCO=[560,0]; R=[28,31,25];

b(2,:)=ones(1,n);%Case\_1 第 1 组

% r=0; m=1; move=[0,23,41,59]; CNCO=[580,0]; R=[30,35,30];

b(2,:)=ones(1,n);%Case\_1 第 2 组

% r=0; m=1; move=[0,18,32,46]; CNCO=[545,0]; R=[27,32,25];

b(2,:)=ones(1,n);%Case\_1 第 3 组

% r=0; m=2; move=[0,20,33,46]; CNCO=[400,378]; R=[28,31,25];

b(2,:)=[2,1,1,1,2,1,2,1];%Case\_2 第 1 组

% r=0; m=2; move=[0,23,41,59]; CNCO=[280,500]; R=[30,35,30];

b(2,:)=[2,1,1,1,1,2,1,2];%Case\_2 第 2 组

% r=0; m=2; move=[0,18,32,46]; CNCO=[455,182]; R=[27,32,25];

b(2,:)=[1,1,1,1,2,2,1,1];%Case\_2 第 3 组

% r=0.01; m=1; move=[0,20,33,46]; CNCO=[560,0]; R=[28,31,25];

b(2,:)=ones(1,n);%Case\_3\_1 第 1 组

% r=0.01; m=1; move=[0,23,41,59]; CNCO=[580,0]; R=[30,35,30];

b(2,:)=ones(1,n);%Case\_3\_1 第 2 组

% r=0.01; m=1; move=[0,18,32,46]; CNCO=[545,0]; R=[27,32,25];

b(2,:)=ones(1,n);%Case\_3\_1 第 3 组

% r=0.01; m=2; move=[0,20,33,46]; CNCO=[400,378]; R=[28,31,25];

b(2,:)=[1,1,1,1,2,2,2,2];%Case\_3\_2 第 1 组

% r=0.01; m=2; move=[0,23,41,59]; CNCO=[280,500]; R=[30,35,30];

b(2,:)=[2,1,1,1,2,1,1,2];%Case\_3\_2 第 2 组

% r=0.01; m=2; move=[0,18,32,46]; CNCO=[455,182]; R=[27,32,25];

b(2,:)=[1,1,1,1,1,2,2,2];%Case\_3\_2 第 3 组

%设定故障率 %设定工序数 %设定移动（分 0,1,2,3 步）耗时 %设定加工时长 %设定

奇偶上下料及清洗时间（3）

%%%%%%%%

hour=8;%设定工作时长

tmin=10;%设定检修下限（单位:分钟）

tmax=20;%设定检修上限（单位:分钟）

N=0;%初始化物料数目

T=0;%初始化时间参量

M=[];%初始化物料记录表

ACC=[];%初始化故障记录表

kACC=0;%初始化故障次数

LR=1;%初始化 RGV 位置信息

b(6,:)=ones(1,n);%故障指示（6）：‘1’正常工作，‘0’故障。

16

bR=0;%初始化 RAG 状态指示

wait=[];%初始化等待队列

kw=n;

wait(:,1)=1:n;

wait(:,2)=zeros(n,1);

ser(:,1)=1:n;

ser(:,2:3)=zeros(n,2);

if m==2

h=[];

kh=0;

end

s=0;%CNC 编号传递中介

dT=0;

while 1

if ~isempty(wait)

for i=1:length(wait(:,1))

wait(i,2)=wait(i,2)+dT;

end

end

%判断是否修好

for i=1:n

if b(6,i)==0

temp0=[];

temp0=find(ACC(:,2)==i);

if T>ACC(temp0(length(temp0)),4)

b(3,i)=0;

b(4,i)=CNCO(1,m)+T-ACC(temp0(length(temp0)),4);

b(6,i)=1;

end

end

end

%更新等待列表

for i=1:n

if (b(1,i)==1)&&(i~=s)&&(b(6,i)==1)

b(4,i)=b(4,i)+dT;

if m==1

if b(4,i)>=CNCO(1,1)

b(1,i)=0;

kw=kw+1;

wait(kw,:)=[i,b(4,i)-CNCO(1,1)];

b(4,i)=0;

b(5,i)=T-wait(kw,2);

end

elseif m==2

b(4,i)=b(4,i)+dT;

17

if b(2,i)==1

if b(4,i)>=CNCO(1,1)

b(1,i)=0;

kw=kw+1;

wait(kw,:)=[i,b(4,i)-CNCO(1,1)];

b(4,i)=0;

b(5,i)=T-wait(kw,2);

if b(3,i)

kh=kh+1;

h(kh,1)=b(3,i);

end

end

elseif b(2,i)==2

if b(4,i)>CNCO(1,2)

b(1,i)=0;

kw=kw+1;

wait(kw,:)=[i,b(4,i)-CNCO(1,2)];

b(4,i)=0;

b(5,i)=T-wait(kw,2);

end

end

end

end

end

%判断时间是否到达

if T>=(hour\*60\*60)

break;

end

%计算服务时间

for i=1:length(wait(:,1))

wait(i,2)=T-b(5,i);

ser(i,1)=wait(i,1);

if m==1

ser(i,2)=R(1,(-mod(ser(i,1),2)+2))+R(1,3);

elseif m==2

if b(2,i)==1

ser(i,2)=R(1,(-mod(ser(i,1),2)+2));

else

ser(i,2)=R(1,(-mod(ser(i,1),2)+2))+R(1,3);

end

%此处根据 m 的数量可继续添加代码

end

end

for i=1:length(ser(:,1))

ser(i,3)=move(1,abs(LR-ceil(ser(i,1)/2))+1);

18

ser(i,2)=ser(i,2)+ser(i,3);

end

dT=0;

%排序&调度开始

p=[];

%排序

if ~isempty(wait)

p(:,1)=wait(:,1);

p(:,2)=(wait(:,2)./ser(:,2))+1;

p(:,3)=ser(:,2);

p(:,4)=ser(:,3);

p=sortrows(p,-2);

%调度

if m==1

LR=ceil(p(1,1)/2);

N=N+1;

M(N,1)=N;

M(N,2)=p(1,1);

M(N,3)=T+p(1,4);

bR=1;

dT=p(1,3);

s=p(1,1);

b(1,p(1,1))=1;

if b(3,p(1,1))

M(b(3,p(1,1)),4)=T+p(1,4);

end

b(3,p(1,1))=N;

b(4,p(1,1))=dT-p(1,4)-R(1,3);

wait(find(wait(:,1)==p(1,1)),:)=[];

ser(find(ser(:,1)==p(1,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(1,1)),temp1,temp2]=acc(r,CNCO(1,1),tmin,tmax);

if ~b(6,p(1,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(1,1);

ACC(kACC,3)=T+p(1,4)+R(1,(-mod(p(1,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(1,3);

elseif m==2

if b(2,p(1,1))==2

if ~isempty(h)

LR=ceil(p(1,1)/2);

19

M(h(1,1),5)=p(1,1);

M(h(1,1),6)=T+p(1,4);

if b(3,p(1,1))

M(b(3,p(1,1)),7)=T+p(1,4);

end

bR=1;

dT=p(1,3);

s=p(1,1);

b(1,p(1,1))=1;

b(3,p(1,1))=h(1,1);

b(4,p(1,1))=dT-p(1,4)-R(1,3);

h(1,:)=[];

kh=kh-1;

wait(find(wait(:,1)==p(1,1)),:)=[];

ser(find(ser(:,1)==p(1,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(1,1)),temp1,temp2]=acc(r,CNCO(1,2),tmin,tmax);

if ~b(6,p(1,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(1,1);

ACC(kACC,3)=T+p(1,4)+R(1,(-mod(p(1,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(1,3);

else

t=0;

for i=2:length(p(:,1))

if b(2,p(i,1))==1

LR=ceil(p(i,1)/2);

N=N+1;

M(N,1)=N;

M(N,2)=p(i,1);

M(N,3)=T+p(i,4);

if b(3,p(i,1))

M(b(3,p(i,1)),4)=T+p(i,4);

end

bR=1;

dT=p(i,3);

s=p(i,1);

b(1,p(i,1))=1;

b(3,p(i,1))=N;

b(4,p(i,1))=dT-p(i,4);

wait(find(wait(:,1)==p(i,1)),:)=[];

20

ser(find(ser(:,1)==p(i,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(i,1)),temp1,temp2]=acc(r,CNCO(1,1),tmin,tmax);

if ~b(6,p(i,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(i,1);

ACC(kACC,3)=T+p(i,4)+R(1,(-

mod(p(i,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(i,3);

t=1;

break

end

if t==0

temp=[];

tempi=0;

for j=1:n

if (b(1,j)==1)&&(b(6,j)==1)

tempi=tempi+1;

temp(1,tempi)=CNCO(1,b(2,j))-b(4,j);

end

end

if ~isempty(temp)

dT=min(temp);

T=T+min(temp);

else

tempp=[];

temppi=0;

for j=1:length(ACC(:,1))

if ACC(j,4)>T

temppi=temppi+1;

tempp(1,temppi)=ACC(j,4)-T;

end

end

if ~isempty(tempp)

dT=min(tempp);

T=T+min(tempp);

end

end

continue;

end

end

21

end

elseif b(2,p(1,1))==1

LR=ceil(p(1,1)/2);

N=N+1;

M(N,1)=N;

M(N,2)=p(1,1);

M(N,3)=T+p(1,4);

bR=1;

dT=p(1,3);

s=p(1,1);

b(1,p(1,1))=1;

if b(3,p(1,1))

M(b(3,p(1,1)),4)=T+p(1,4);

end

b(3,p(1,1))=N;

b(4,p(1,1))=dT-p(1,4)-R(1,3);

wait(find(wait(:,1)==p(1,1)),:)=[];

ser(find(ser(:,1)==p(1,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(1,1)),temp1,temp2]=acc(r,CNCO(1,1),tmin,tmax);

if ~b(6,p(1,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(1,1);

ACC(kACC,3)=T+p(1,4)+R(1,(-mod(p(1,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(1,3);

end

end

else

temp=[];

tempi=0;

for i=1:n

if (b(1,i)==1)&&(b(6,i)==1)

tempi=tempi+1;

temp(1,tempi)=CNCO(1,b(2,i))-b(4,i);

end

end

if ~isempty(temp)

dT=min(temp);

T=T+min(temp);

else

tempp=[];

22

temppi=0;

for j=1:length(ACC(:,1))

if ACC(j,4)>T

temppi=temppi+1;

tempp(1,temppi)=ACC(j,4)-T;

end

end

if ~isempty(tempp)

dT=min(tempp);

T=T+min(tempp);

end

end

continue;

end

end

%计算加工零件个数

CN=0;

for i=1:length(M(:,1))

if (M(i,3\*m+1))||(((60\*60\*hour-M(i,3\*m))>=CNCO(1,m))&&(M(i,3\*m)~=0))

CN=CN+1;

end

end

CN=CN-kACC;

fprintf('%d\t 小时加工物料个数为\t%d\n',hour,CN);

2、仿真函数

function [CN]=manage0(b0)

n=8;%设定 CNC 数目

if b0(1)&&b0(2)&&b0(3)&&b0(4)&&b0(5)&&b0(6)&&b0(7)&&b0(8)

m=2;%设定物料工序

%%%%参数修改区

r=0; move=[0,20,33,46]; CNCO=[400,378]; R=[28,31,25]; %Case\_2 第 1 组

% r=0; move=[0,23,41,59]; CNCO=[280,500]; R=[30,35,30]; %Case\_2 第 2 组

% r=0; move=[0,18,32,46]; CNCO=[455,182]; R=[27,32,25]; %Case\_2 第 3 组

% r=0.01; move=[0,20,33,46]; CNCO=[400,378]; R=[28,31,25]; %Case\_3\_2 第 1

组

% r=0.01; move=[0,23,41,59]; CNCO=[280,500]; R=[30,35,30]; %Case\_3\_2 第 2

组

% r=0.01; move=[0,18,32,46]; CNCO=[455,182]; R=[27,32,25]; %Case\_3\_2 第 3

组

%设定故障率 %设定移动（分 0,1,2,3 步）耗时 %设定加工时长 %设定奇偶上下料

及清洗时间（3）

%%%%

hour=8;%设定工作时长

tmin=10;%设定检修下限（单位:分钟）

23

tmax=20;%设定检修上限（单位:分钟）

N=0;%初始化物料数目

T=0;%初始化时间参量

M=[];%初始化物料记录表

ACC=[];%初始化故障记录表

kACC=0;%初始化故障次数

LR=1;%初始化 RGV 位置信息

b=zeros(5,n);%初始化 CNC 状态指示（1）&物料指示（3）&时间指示（4）&停机

指示（5）

for i=1:n

b(2,i)=b0(i);

end%类型指示（2）

b(6,:)=ones(1,n);%故障指示（6）：‘1’正常工作，‘0’故障。

bR=0;%初始化 RAG 状态指示

wait=[];%初始化等待队列

kw=n;

wait(:,1)=1:n;

wait(:,2)=zeros(n,1);

ser(:,1)=1:n;

ser(:,2:3)=zeros(n,2);

t0=clock;

if m==2

h=[];

kh=0;

end

s=0;%CNC 编号传递中介

dT=0;

while 1

if ~isempty(wait)

for i=1:length(wait(:,1))

wait(i,2)=wait(i,2)+dT;

end

end

if etime(clock,t0)>2

break;

end

%判断是否修好

for i=1:n

if b(6,i)==0

temp0=[];

temp0=find(ACC(:,2)==i);

if T>ACC(temp0(length(temp0)),4)

b(3,i)=0;

b(4,i)=CNCO(1,m)+T-ACC(temp0(length(temp0)),4);

b(6,i)=1;

24

end

end

end

%更新等待列表

for i=1:n

if (b(1,i)==1)&&(i~=s)&&(b(6,i)==1)

b(4,i)=b(4,i)+dT;

if m==1

if b(4,i)>=CNCO(1,1)

b(1,i)=0;

kw=kw+1;

wait(kw,:)=[i,b(4,i)-CNCO(1,1)];

b(4,i)=0;

b(5,i)=T-wait(kw,2);

end

elseif m==2

b(4,i)=b(4,i)+dT;

if b(2,i)==1

if b(4,i)>=CNCO(1,1)

b(1,i)=0;

kw=kw+1;

wait(kw,:)=[i,b(4,i)-CNCO(1,1)];

b(4,i)=0;

b(5,i)=T-wait(kw,2);

if b(3,i)

kh=kh+1;

h(kh,1)=b(3,i);

end

end

elseif b(2,i)==2

if b(4,i)>CNCO(1,2)

b(1,i)=0;

kw=kw+1;

wait(kw,:)=[i,b(4,i)-CNCO(1,2)];

b(4,i)=0;

b(5,i)=T-wait(kw,2);

end

end

end

end

end

%判断时间是否到达

if T>=(hour\*60\*60)

break;

end

25

%计算服务时间

for i=1:length(wait(:,1))

wait(i,2)=T-b(5,i);

ser(i,1)=wait(i,1);

if m==1

ser(i,2)=R(1,(-mod(ser(i,1),2)+2))+R(1,3);

elseif m==2

if b(2,i)==1

ser(i,2)=R(1,(-mod(ser(i,1),2)+2));

else

ser(i,2)=R(1,(-mod(ser(i,1),2)+2))+R(1,3);

end

%此处根据 m 的数量可继续添加代码

end

end

for i=1:length(ser(:,1))

ser(i,3)=move(1,abs(LR-ceil(ser(i,1)/2))+1);

ser(i,2)=ser(i,2)+ser(i,3);

end

dT=0;

%排序&调度开始

p=[];

%排序

if ~isempty(wait)

p(:,1)=wait(:,1);

p(:,2)=(wait(:,2)./ser(:,2))+1;

p(:,3)=ser(:,2);

p(:,4)=ser(:,3);

p=sortrows(p,-2);

%调度

if m==1

LR=ceil(p(1,1)/2);

N=N+1;

M(N,1)=N;

M(N,2)=p(1,1);

M(N,3)=T+p(1,4);

bR=1;

dT=p(1,3);

s=p(1,1);

b(1,p(1,1))=1;

if b(3,p(1,1))

M(b(3,p(1,1)),4)=T+p(1,4);

end

b(3,p(1,1))=N;

b(4,p(1,1))=dT-p(1,4)-R(1,3);

26

wait(find(wait(:,1)==p(1,1)),:)=[];

ser(find(ser(:,1)==p(1,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(1,1)),temp1,temp2]=acc(r,CNCO(1,1),tmin,tmax);

if ~b(6,p(1,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(1,1);

ACC(kACC,3)=T+p(1,4)+R(1,(-mod(p(1,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(1,3);

elseif m==2

if b(2,p(1,1))==2

if ~isempty(h)

LR=ceil(p(1,1)/2);

M(h(1,1),5)=p(1,1);

M(h(1,1),6)=T+p(1,4);

if b(3,p(1,1))

M(b(3,p(1,1)),7)=T+p(1,4);

end

bR=1;

dT=p(1,3);

s=p(1,1);

b(1,p(1,1))=1;

b(3,p(1,1))=h(1,1);

b(4,p(1,1))=dT-p(1,4)-R(1,3);

h(1,:)=[];

kh=kh-1;

wait(find(wait(:,1)==p(1,1)),:)=[];

ser(find(ser(:,1)==p(1,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(1,1)),temp1,temp2]=acc(r,CNCO(1,2),tmin,tmax);

if ~b(6,p(1,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(1,1);

ACC(kACC,3)=T+p(1,4)+R(1,(-mod(p(1,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(1,3);

else

t=0;

27

for i=2:length(p(:,1))

if b(2,p(i,1))==1

LR=ceil(p(i,1)/2);

N=N+1;

M(N,1)=N;

M(N,2)=p(i,1);

M(N,3)=T+p(i,4);

if b(3,p(i,1))

M(b(3,p(i,1)),4)=T+p(i,4);

end

bR=1;

dT=p(i,3);

s=p(i,1);

b(1,p(i,1))=1;

b(3,p(i,1))=N;

b(4,p(i,1))=dT-p(i,4);

wait(find(wait(:,1)==p(i,1)),:)=[];

ser(find(ser(:,1)==p(i,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(i,1)),temp1,temp2]=acc(r,CNCO(1,1),tmin,tmax);

if ~b(6,p(i,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(i,1);

ACC(kACC,3)=T+p(i,4)+R(1,(-

mod(p(i,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

end

T=T+p(i,3);

t=1;

break

end

if t==0

temp=[];

tempi=0;

for j=1:n

if (b(1,j)==1)&&(b(6,j)==1)

tempi=tempi+1;

temp(1,tempi)=abs(CNCO(1,b(2,j))-b(4,j));

end

end

if ~isempty(temp)

dT=min(temp);

28

T=T+min(temp);

else

tempp=[];

temppi=0;

for j=1:length(ACC(:,1))

if ACC(j,4)>T

temppi=temppi+1;

tempp(1,temppi)=ACC(j,4)-T;

end

end

if ~isempty(tempp)

dT=min(tempp);

T=T+min(tempp);

end

end

continue;

end

end

end

elseif b(2,p(1,1))==1

LR=ceil(p(1,1)/2);

N=N+1;

M(N,1)=N;

M(N,2)=p(1,1);

M(N,3)=T+p(1,4);

bR=1;

dT=p(1,3);

s=p(1,1);

b(1,p(1,1))=1;

if b(3,p(1,1))

M(b(3,p(1,1)),4)=T+p(1,4);

end

b(3,p(1,1))=N;

b(4,p(1,1))=dT-p(1,4)-R(1,3);

wait(find(wait(:,1)==p(1,1)),:)=[];

ser(find(ser(:,1)==p(1,1)),:)=[];

kw=kw-1;

temp1=0;temp2=0;

[b(6,p(1,1)),temp1,temp2]=acc(r,CNCO(1,1),tmin,tmax);

if ~b(6,p(1,1))

kACC=kACC+1;

ACC(kACC,1)=kACC;

ACC(kACC,2)=p(1,1);

ACC(kACC,3)=T+p(1,4)+R(1,(-mod(p(1,1),2)+2))+temp1;

ACC(kACC,4)=ACC(kACC,3)+temp2;

29

end

T=T+p(1,3);

end

end

else

temp=[];

tempi=0;

for i=1:n

if (b(1,i)==1)&&(b(6,i)==1)

tempi=tempi+1;

temp(1,tempi)=abs(CNCO(1,b(2,i))-b(4,i));

end

end

if ~isempty(temp)

dT=min(temp);

T=T+min(temp);

else

tempp=[];

temppi=0;

for j=1:length(ACC(:,1))

if ACC(j,4)>T

temppi=temppi+1;

tempp(1,temppi)=ACC(j,4)-T;

end

end

if ~isempty(tempp)

dT=min(tempp);

T=T+min(tempp);

end

end

continue;

end

end

%计算加工零件个数

CN=0;

for i=1:length(M(:,1))

if (M(i,3\*m+1))||(((60\*60\*hour-M(i,3\*m))>=CNCO(1,m))&&(M(i,3\*m)~=0))

CN=CN+1;

end

end

CN=CN-kACC;

else

CN=0;

end

end

30

3、模拟故障判定函数

function [y1,y2,y3]=acc(r,tr,t1,t2)

y1=1;

y2=0;

y3=0;

while 1

g=rand(1,1);

if g~=0

if g<=r

y1=0;

end

break;

end

end

if y1==0

y2=round(tr\*rand(1,1));

while 1

y3=normrnd((t1+t2)/2,abs((t2-t1)/2));

if y3>=10&&y3<20

y3=round(60\*y3);

break;

end

end

end

end

4、遗传算法程序

clear

n=8;

m=2;

%%%%%参数修改区

Z0=100;

G=10;

p1=0.5;

p2=0.25;

%%%%%

Z=[];

kZ=0;

while 1

if kZ>=Z0

break;

end

kZ=kZ+1;

for i=1:n

31

Z(kZ,i)=randperm(m,1);

end

try

Z(kZ,n+1)=manage0(Z(kZ,1:n));

catch ErrorInfo

Z(kZ,:)=[];

kZ=kZ-1;

end

end

Z=sortrows(Z,-(n+1));

for ii=1:G

for i=1:2:Z0

node=zeros(1,n);

for j=1:n

if randperm(2,1)==1

node(j)=1;

end

end

v1=zeros(1,n);

v2=v1;

for j=1:n

if node(j)==0

v1(j)=Z(i,j);

v2(j)=Z(i+1,j);

else

v1(j)=Z(i+1,j);

v2(j)=Z(i,j);

end

end

try

v1(n+1)=manage0(v1(1:n));

kZ=kZ+1;

Z(kZ,:)=v1;

catch ErrorInfo

end

try

v2(n+1)=manage0(v2(1:n));

kZ=kZ+1;

Z(kZ,:)=v1;

catch ErrorInfo

end

end

Z=sortrows(Z,-(n+1));

tempp=length(Z(:,1));

i=floor(Z0/2);

32

while 1

for j=1:n

if (rand(1,1)<p2)

Z(i,j)=randperm(m,1);

end

end

try

Z(i,n+1)=manage0(Z(kZ,1:n));

i=i+1;

catch ErrorInfo

Z(i,:)=[];

kZ=kZ-1;

end

if i>length(Z(:,1))

break;

end

end

Z=sortrows(Z,-(n+1));

temp=length(Z(:,1));

while 1

if length(Z(:,1))>Z0

Z(kZ,:)=[];

kZ=kZ-1;

else

break;

end

end

end

fprintf('第%d 代最优组合为\t（%d',G,Z(1,1));

for i=2:n

fprintf(',%d',Z(1,i));

end

fprintf(')\n');

33