基于改进遗传算法的自动车调度策略及其优化

摘要

随着“工业 4.0”与“中国制造 2025”等概念的提出，中国制造业对于现代化、智能化、自动化的生产加工系统越发重视。轨道式自动引导车(Rail Guide Vehicle, RGV)作为现代化、智能化、自动化的制造运输系统之一正在应用到越来越多的生产制造项目当中。在 RGV 的生产应用中，如何有效地调动使之发挥最大的效率是非常有应用价值的课题。

本文主要采用改进的遗传算法，首先建立了一般情况下的 RGV 调动基本模型。对于一道工序的调度策略，本文采用一维数字基因编码描述调度策略的方式；对于两道工序的调度，首先使用层次分析法得出最佳工序分配方案，其次使用二维基因编码建立遗传算法模型；对于考虑故障的情况，采用随机动态模拟方法引入故障的影响。其后，提出多目标优化、结合模拟退火算法防止早熟、改进初始种群的方法，基于原始模型进行优化；然后，使用三组实际参数求解模型，得到四种情况下最优 RGV 调度策略；最后，对模型进行收敛分析、软件模拟、敏感度分析等方法评估了改进模型的实用性和有效性。

针对一道工序且不考虑故障的工作模式，本文提出了单目标优化的模型，将一个班次内加工的物料成品数目作为目标函数，借助遗传算法搜索寻优，避免了朴素搜索模型时间复杂度过高的问题。在遗传算法中，本文首先使用 CNC 的序号进行染色体编码，能够表示可行解域中的所有策略；然后采用精英保留策略与轮盘赌方法相结合的选择方式，避免基因缺失，提高计算效率；最后求解遗传算法模型，得到最优调度方案，最大化提高了第一种模式的工作效率。

对于两道工序且不考虑故障的工作模式，首先运用层次分析法，根据第一、第二道工序时间成本确定每一台 CNC 的工序；然后采用二维基因编码的方法描述两道工序时的调度策略，建立遗传算法模型；最后使用与一道工序相同的求解方法，得到最优调度方案。

对于考虑故障的工作模式，本文首先确定了单位时间的平均故障发生次数，以此建立动态随机故障模拟模型；其次，将动态故障模拟模型与遗传算法模型结合，得到考虑故障的两种工序求解模型；最后，对于两种工序模式，分别采用结合动态故障模拟的模型求解，得到最优化调度方案。

建立基础模型后，本文从提高系统效率、防止“早熟收敛”、改进寻优方法三个方面改进原模型，得到改进后的 RGV 调度模型。其中使用多目标优化的方式提高系统运行效率；将模拟退火算法结合到交换过程防止“早熟收敛”；调整初始种群产生方法提升了模型寻优能力。

最后，本文对改进模型使用在线性能评估、软件算法模拟、敏感度检测、将结果表达为甘特图等方法，检验了模型的实用性和算法的有效性与稳定性。

关键词： RGV 调度策略，改进的遗传算法，层次分析法，模拟退火算法，动态模拟

一. 问题重述

问题的应用场景是由 8 台计算机数控机床 (Computer Number Controller, CNC)、一辆轨道式自动引导车，一条直线轨道和上下料传送带等附属设备组成的智能加工系统。此系统的加工作业情况有三种：第一种为加工一道工序的物料，此时每一台 CNC 使用相同刀具，加工可以使用任意一台 CNC；第二种情况是加工两道工序的物料，每个物料的第一和第二道工序分别由两台不同的 CNC 加工，此时每一台 CNC 只能安装第一或第二道工序使用的刀具，在作业过程中不可更换；第三种情况考虑 CNC 在加工过程中发生故障的可能。其中每台 CNC 在加工过程中发生故障的概率为 1%，修复时间需要 10~20 分钟，故障中未完成的物料报废。在第三种情况中分别考虑加工一道和两道工序物料的情况。

首先，任务一要求在一般应用场景下针对三种情况给出 RGV 的调度模型和相应的求解算法。然后，任务二给出了三组系统作业参数，要求使用三组参数分别计算出情况一、情况二、考虑发生故障的情况一、考虑发生故障的情况二，共四个应用场景下系统连续工作一个班次（ 8 小时）的调度情况，并且给出 RGV 调度策略和系统的作业效率，以检验模型的实用性和算法的有效性。

二. 问题分析

本问题分为两个部分：建立一般 RGV 调度模型与求解算法、将参数带入模型得到一个班次中 RGV 调度过程以检验算法有效性并评估效率。其中每个部分模型都包含四种情况：

（1）加工一道工序的物料，不考虑发生故障；
（2）加工两道工序的物料，不考虑发生故障；
（3）加工一道工序的物料，CNC 在加工过程中有 1%的可能性发生故障;
（4）加工两道工序的物料，CNC 在加工过程中有 1%的可能性发生故障。

2.1 加工一道工序的物料，不考虑发生故障

第一个问题是在不考虑故障的条件下设计加工一道工序物料的调度方法。其作业流程如下: 首先加工系统通电启动后，RGV 初始位置在一号 CNC 与二号 CNC 之间，所有 CNC 都处于空闲状态；然后空闲状态的 CNC 向 RGV 发出上料信号，由 RGV 自行确定 CNC 的上下料作业次序；上下料作业之后进行清洗作业，具体步骤包括三步：一只机械手爪从清洗槽取出成料、另一只机械手将刚取下的熟料放入清洗槽、机械臂转动将成料放上传送带。完成一项作业后立即判别下一个指令，若无指令则等待；一个班次结束之后作业停止，RGV 回到初始位置。此作业流程用流程图表示如下图所示。

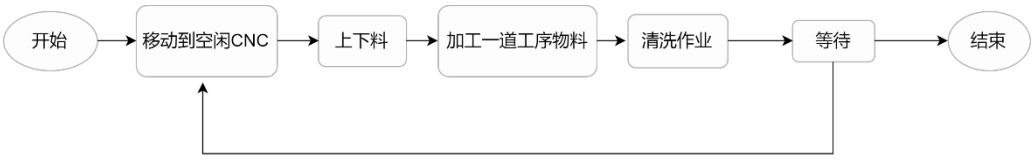


图 1 加工一道工序物料的流程图

我们的目标是在这个作业流程的约束下，尽量提高调度的效率。从数学规划的角度来看，这个问题可表达为在约束条件下实现一个或多个函数的优化[]。具体来说，就是合理地安排作业的加工次序和开始时间，在满足约束条件的情况下优化一些性能指标。本题的应用场景为单台 RGV，以调度性能为指标的静态调度问题。随着调度理论研究的深入和交叉学科算法的出现，目前涌现出了许多有效的调度方法，比如模拟退火，神经网络与遗传算法等[]。

在这些方法中，遗传算法是一种强有力的随机搜索优化算法，它的应用范围十分广泛，能够解决一些传统方法很难解决的问题，而且相对与其他现代方法收敛较快，容易达到全局最优点。然而简单的遗传算法往往不能很好地解决这类工程问题，因此本题采用改进的遗传算法建立调度模型。在应用遗传算法时非常重要的一点是如何用基因编码来描述调度过程。由于一共有 8 个 CNC，且每次 RGV 可在任意一个 CNC 处加工材料，因此我们采用 1-8 编码一维序列作为基因型描述 RGV 的调动顺序。

2.2 加工两道工序的物料，不考虑发生故障；

第二个问题是在不考虑故障的条件下设计加工两道工序物料的调度方法。其基本约束条件与第一道工序类似，特殊条件在于第一和第二道工序需要在不同的两台 CNC 上依次完成，所用时间也不同。由于每台 CNC 只能安装一种加工刀片且作业过程中不能更换，所以每台 CNC 只能进行第一或第二道工序。相比一道工序物料加工过程，两道工序物料的加工中间步骤多了几步，其流程图如下所示。

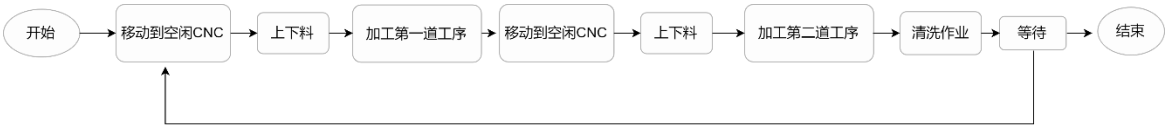


图 2 加工两道工序物料流程图

由于每天 CNC 只能进行一种加工，因此在使用遗传算法优化之前需要确定 CNC 的分配策略，即哪些 CNC 执行第一道工序，哪些执行第二道工序。确定之后使用遗传算法进行搜索优化，在使用遗传算法时考虑到因为执行第一道或第二道工序的 CNC 已经固定，所以一维基因编码方法不能很好地表达出两道工序的组合，因此采用二维的基因编码方法描述工序流程。

2.3 加工一道工序的物料，CNC 在加工过程中有 1%的可能性发生故障

第三个问题是在第一种情况的条件下增加对故障发生可能性的考虑，即加工一道工序的物料，同时考虑发生故障的可能性。其中系统作业流程完全相同，但是 CNC 在加工过程中有 1%的可能发生故障。发生故障之后需要人工处理，时间在 10~20 分钟之间。CNC 发生故障时正在处理的物料报废，并且需要时间处理，这里我们取处理时间平均数 15 分钟。

建立加工一道工序物料的调度模型并考虑可能发生故障，要先解决两个问题，首先是如何根据 1%的故障可能性确定可能发生故障的 CNC，然后解决如何计算由于故障影响导致的时间延长。由于第三个问题与第一个问题的执行流程相同，所以在采用遗传算法的过程中可以采用第一个问题中的基因编码方式用来描述调度方案过程。

2.4 加工两道工序的物料，CNC 在加工过程中有 1%的可能性发生故障

第四个问题是在第二种情况的条件下增加对故障发生可能性的考虑，也就是加工两道工序的物料同时考虑发生故障的可能性。基于第二和第三个问题的讨论，我们用相同的方法确定发生故障的 CNC，并且同样采用二维基因编码方式描述调度方案过程。

三. 模型假设

1. 假设一：在加工过程中只有前一道工序完成才会进行下一道；
2. 假设二：RGV 在移动过程中 CNC 不会产生故障；
3. 假设三：人工处理故障 CNC 的时间平均为 15 分钟

四. 符号说明

符号	符号说明	符号	符号说明
M	种群大小	$\mathbf{A}$	判断矩阵
G	进化最大代数	$\boldsymbol{\omega}$	权值向量
p_c	染色体交叉概率	λ	矩阵特征值
p_m	基因变异概率	CI	随机一致性指标
T_i	CNC 上加工开始时间	CR	一致性比率
ω, γ, β	基因编码码元	λ_p	泊松分布事件到达率
$Fitness, F$	适应性函数	E_i	模拟退火中状态 i 能量
num	一个班次加工物料数	$P_{i,s}$	轮盘赌被选中概率

五. 建立一般 RGV 调度模型

5.1 加工一道工序的物料，不考虑发生故障

5.1.1 基础遗传算法模型的建立方法

遗传算法 (Genetic Algorithm, GA) 是一种模拟自然界生命进化机制的搜索寻优算法。这种算法的原理是基于自然选择和遗传机制，比如基因交换与变异机制。这种算法在人工系统中实现特定目标的优化，实质是通过群体搜索，根据适者生存的原则逐代淘汰某些个体实现群体进化，最终得到最优解[1]。在 RGV 调度问题中，我们的目标是尽可能提高系统的效率：在八小时的单个班次中，尽可能多地加工物料。遗传算法的过程如下：

（1）根据规则产生出事种群，求每一个个体的适应度。

（2）根据适者生存原则淘汰掉适应度较低的个体，选择出优良个体；

（3）被选出的优良个体两两配对，根据遗传和变异规则产生后代；

（4）第二代种群重复过程 (1) – (3) 的进化过程直到满足进化终止条件；

（5）进化过程终止之后即可根据适应度确定最优个体，即最优方案。

实现遗传算法模型的方法中要确定一些参数和规则，这些参数影响着进化过程的方向和快慢。模型的实现方法如下：

（1）根据具体问题确定约束条件和可行解域，确定一种能够表示可行解域每一解的基因编码方式。

（2）确定适应度函数。适应度函数既规定了种群进化的方向，其函数值又是淘汰个体的依据，因此适应度函数要能够反映模型的优化目的，并且有合理的结构。

（3）确定进化参数：种群规模M、染色体随即交叉概率p_c、基因变异概率p_m，以及进化终止条件。

5.1.2 遗传算法进化参数确定

GA 中需要选择的进化参数主要有全体大小，进化最大代数，交叉概率，变异概率等。求解的遗传算法参数设定如下:

参数名称	参数符号	参数值
种群大小	M	800
进化最大代数	G	100
染色体交叉概率	p_c	1
基因变异概率	p_m	0.001

表 1 遗传算法进化参数

（1）种群大小 M

种群大小 M 表示了每一代群体中含有的个体数目。当 M 较小时可提高遗传算法的运算速度，但会使种群多样性降低，有可能造成 GA 的“早熟”现象，即收敛于局部最优点；当 M 较大时会降低运行速度。经过尝试与调整在调度模型中种群大小 M 取值为 800。

（2）进化最大代数 G

此调度模型将进化最大代数作为收敛的停止条件，达到最大代数以后算法停止。在本模型中进化最大代数 G 取值 100 代，在实际运算中可以保证收敛。此参数的取值较为灵活，可根据保证收敛的最小进化代数来确定。

（3）染色体交叉概率p_c

交叉操作是遗传算法产生新个体编码的主要方式之一，交叉概率一般取值应该较大。若取值过小则产生新个体的速度较慢，导致收敛所需的进化代数增大，收敛较慢；但若取值过大又会导致种群中某些优良的特性被破坏。经过实际测试，在此模型中染色体交叉概率取值为 1。

（4）基因变异概率p_m

若变异概率取值较大，虽然可以产生新的编码模式，增大种群的基因多样性，但若变异概率取值过大也会导致某些优良特性被破坏，则有可能导致“早熟现象”。根据 Davis[2]的建议，取值一般在 0.0001~0.1000。

5.1.3 确定约束条件

（1）顺序约束：相邻工件的加工顺序约束

$$T_i - T_j \geq m_i, \qquad 1 \leq \mathrm{i,j} \leq \mathrm{n}$$

其中T_i表示在第i个 CNC 上加工开始的时间，T_j表示在第j个 CNC 上加工开始的时间,

m_i表示在第 i 个 CNC 上加工需要的时间，并且ω_i, ω_j在编码中出去相邻位置，也就是说 KGV 在第i个 CNC 上加工结束之后即前往第 j 个 CNC 处加工下一物料。此式描述的是相邻两个物料加工要在前一个结束后下一个才开始。

（2）资源约束：同一台 CNC 在当前加工任务结束后才能开始另一个任务的加工

$$T_{i,k} - T_{j,k} \geq m_{i,k}, \quad 1 \leq \mathrm{i,j} \leq \mathrm{n}, 1 \leq \mathrm{k} \leq 8$$

其中$T_{i,k}, T_{j,k}$表示相邻物料在编号为 k 的 CNC 上加工开始的时间，$m_{i,k}$表示编号为 i 的物料在编号为 k 的 CNC 上加工的时间长度；这个约束条件表示，在某一个确定的时间点，同一个 CNC 不能同时加工任意两个不同的物料。

5.1.4 基因编码方式

在加工一道工序的物料工作中，由于一共有八台 CNC 且分别编码为 1-8 号，同时加工每个物料都可以选择任意一台 CNC，所以为了描述自动车调度过程，采用如下编码方式：

$$\omega_1, \omega_2, \omega_3, \omega_4, \ldots \omega_n, \quad \omega_i \in \{1,2,3,4,5,6,7,8\}$$

其中每一个ω_i表示 RGA 在第ω_i个 CNC 处加工物料，ω_n表示当 RGA 在第ω_n个 CNC 处完成加工物料后，一个班次（8 小时）已经结束。在这里 n 为固定值，为了使各代每个个体染色体编码长度相等。因此，这一编码序列表示在固定的一段时间内，RGA 从初始位置出发，首先移动到ω_1上下料、加工、清洗，然后移动到ω_2进行相同的操作，然后移动到$\omega_3, \omega_4, \ldots, \omega_n$，最后回到初始位置的过程。由此可知此基因编码方式可以描述一种调度策略。

需要说明的是，根据假设条件 1，如果在ω_i处结束加工，正要到ω_{i+1}处去，但是此时ω_{i+1}并非空闲，那么此时在原地等待，即插入一段等待时间。

5.1.5 确定适应性函数

适应性函数 (Fitness Function) 是衡量每一代种群中个体适应度的函数，也是我们的目标函数。在进行自然选择时首先对当前代种群每一个个体根据适应性函数求其适应性值，然后淘汰掉适应性差的个体。因此进化的过程也就是不断通过遗传变异和自然选择搜索出适应性函数值最高个体（基因编码），在 RGV 调度策略问题中也就是寻找到在固定时间内加工物料数最多的调度策略。

根据这一目标，我们确定适应性函数的表达式为：

$$Fitness = num - p$$

其中$Fitness$表示个体的适应性，num表示在一个班次（8 小时）内加工完成的物料总数；p为一个常数，它的选择标准低于整体num数值的一个常数。这里加入这个参数的目的是突出不同个体适应度的相对差距，有利于自然选择。

5.1.6 选择策略

遗传算法的基本原理是自然选择原则。选择在计算出当前种群每个个体的适应性函数值之后进行。选择的目的是避免基因缺失，提高计算效率和加速收敛。此模型中采用的选择策略为轮盘赌(Roulette Wheel)选择与精英保留策略(Elitist Strategy)的结合。

轮盘赌选择的基本思想是每个个体被选择的概率与其适应度大小成正比。当群体

大小为 M，个体适应度为F_i时，个体 i 被选中的概率$P_{i,s}$为：

$$P_{i,s} = \frac{F_i}{\sum_{i=1}^{M} F_i},\ (i = 1,2,3,\dots,M)$$

轮盘赌使得适应度高的个体有更大的概率被选中，也就是适应度越高，基因编码越容易被保留。因此这种选择方法有助于加快收敛。

精英保留策略的核心思想是在每一次产生新的一代时，首先把当前适应度高的“精英基因”复制到新的一代中。精英保留策略是基本遗传算法的一种优化，它防止进化过程中产生的最优解被交叉和变异所破坏，因此可以大幅提高运算速度，加快收敛。

本调度模型将精英保留策略与轮盘赌两种选择策略结合，具体的选择方法如下：

（1）计算当前种群中所有个体的适应度函数值，并按照降序排列；

（2）保留适应度函数前 50%的个体，将它们的基因直接保留作为一部分子代，后 50%个体舍弃；

（3）在保留的基因型中使用轮盘赌法计算出每个个体被选中的概率，然后选出 0.5M 对个体作为亲本交配产生子代，每对亲本产生一个子代，得到的 0.5M 个基因编码作为另一部分子代；

（4）将按照精英保留策略选中的 0.5M 个基因编码与轮盘赌产生的 0.5M 个基因编码合并得到下一代种群。

5.1.7 交叉和变异

（1）交叉操作

交叉操作采用单点交叉。在经过选择过程之后根据交叉概率p_c选出一部分亲本进行交叉操作，若经过选择之后得到的两个亲本基因编码为：

$$f_1 = [\omega_1, \omega_2, \omega_3, \dots., \omega_n]$$
$$f_2 = [\gamma_1, \gamma_2, \gamma_3, \dots., \gamma_n]$$

产生一个随机数 s, 满足$1 \le s \le n$, 将第 s 个基因作为交叉点，经过交叉运算得到的子代个体为t_1, t_2，则它们的基因型为：

$$t_1 = [\omega_1, \omega_2, \dots \omega_s, \gamma_{s+1}, \dots, \gamma_n]$$
$$t_2 = [\gamma_1, \gamma_2, \dots \gamma_s, \omega_{s+1}, \dots, \omega_n]$$

至此完成了染色体单点交叉操作。

（2）变异操作

变异是以很小的概率p_m随机改变染色体上的某些位从而得到新的个体。变异是实现群体多样性的一种手段，也是收敛到全局最优点的保证。本模型中采用的变异方法如下：

首先按照变异率选定变异个体，然后随机选取三个整数 u, v, w 并且满足

$$1 < u < v < w < n$$

然后把 u, v 之间并且包括 u, v 两个基因的基因段插入到 w 后面去。

以上是加工需要一道工序的物料且不考虑发生故障时的 RGV 调度策略模型。总结来看，模型首先确定了约束条件和编码策略，根据策略确定进化参数；然后随机产生初始种群，以固定时间加工物料数目作为适应性函数；选择策略使用轮盘赌与精英保留策略，同时使用单点交叉和变异的方法扩大基因多样性和加快收敛，建立了此调度模型。

5.2 加工两道工序的物料，不考虑发生故障

如问题分析中所阐述，加工两道工序物料与一道工序的关键不同在于每个 CNC 只能进行一道工序的处理，因此每次 RGV 要将同一个物料送至两个不同的 CNC 进行加工。建立两道工序的模型同样采用遗传算法，但是与一道工序有两点重要的不同：确定每个 CNC 执行哪一道工序和编码策略。

5.2.1 使用层次分析法确定各 CNC 执行工序

由于 CNC 在每个班次内只能进行加工一道工序，而且参数不同加工第一道或第二道工序所花的时间不同，奇数编号和偶数编号的 CNC 上下料的时间也不同。为了综合考虑这些因素，我们建立如下图所示层次分析模型分析每一个 CNC 应该执行第几道工序。

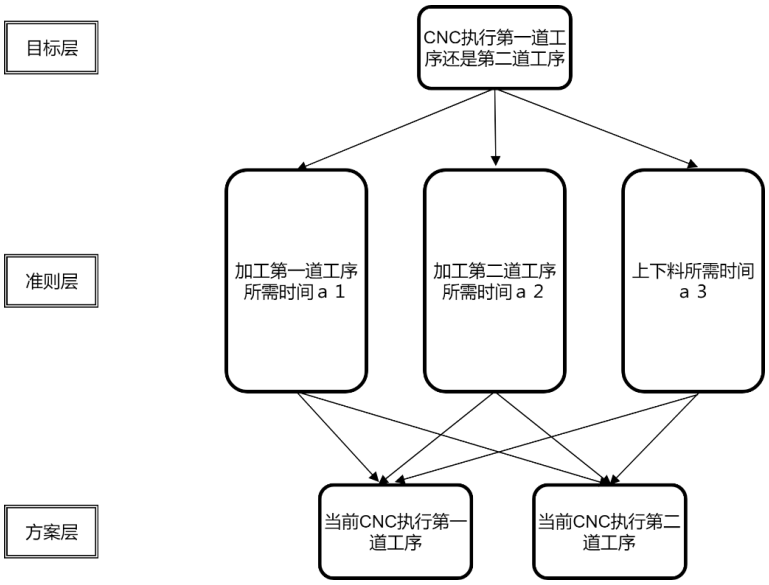


图 3 确定 CNC 加工工序的层次结构

图中a_1, a_2, a_3分别为某个 CNC 加工第一道、第二道工序所需时间和上下料所需时间，当使用第几组参数和 CNC 编码确定之后就可以确定具体值。

（1）构造准测层对目标层的一致性矩阵

一般的层次分析法采用 9 级标度法通过评分构造一致性矩阵，然而本模型使用加工第一道工序、第二道工序、上下料时间之间的比值构造一致性矩阵，更能反应客观情况，找到时间上最优化的安排方案。于是构造一致性矩阵：

$$\mathbf{A}=\begin{bmatrix} 1 & \frac{a_2}{a_1} & \frac{a_3}{a_1} \\ \frac{a_1}{a_2} & 1 & \frac{a_3}{a_2} \\ \frac{a_1}{a_3} & \frac{a_2}{a_3} & 1 \end{bmatrix}$$

一致性矩阵的任意列向量都是特征向量，而矩阵 A 作为正互反矩阵，因此它的一致性比较好，它的列向量应该为近似特征向量，所以可取其列向量的平均作为特征向量的近似解。

首先，我们按列向量对一致性矩阵进行归一化，然后按行相加再归一化得到特

征向量$\boldsymbol{\omega}$。假设记按列向量归一化之后的矩阵 A 为$\widehat{\mathbf{A}}$：

$$\widehat{\mathbf{A}}=\begin{bmatrix}\alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3}\\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3}\\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3}\end{bmatrix}$$

则按行相加再归一化得到的特征向量$\boldsymbol{\omega}$的表达式为：

$$\boldsymbol{\omega}=\begin{pmatrix}\frac{\sum_{i=1}^{3}\alpha_{1,i}}{\sum_{i=1}^{3}\sum_{j=1}^{3}\alpha_{i,j}}\\ \frac{\sum_{i=1}^{3}\alpha_{2,i}}{\sum_{i=1}^{3}\sum_{j=1}^{3}\alpha_{i,j}}\\ \frac{\sum_{i=1}^{3}\alpha_{3,i}}{\sum_{i=1}^{3}\sum_{j=1}^{3}\alpha_{i,j}}\end{pmatrix}=\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\end{pmatrix}$$

根据特征向量求解得到特征值λ, 接下来计算一致性指标：

$$\mathrm{CI}=\frac{\lambda-n}{n-1}$$

其中 n 为 A 矩阵的维数，根据维数查询随机一致性指标 RI，表为如下：

n	1	2	3	4	5	6	7	8
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41

表 2 随机一致性指标 RI 查找表

则一致性比率指标：

$$\mathrm{CR}=\frac{\mathrm{CI}}{\mathrm{RI}}$$

若 CR<0.1 则矩阵的一致性可以接受，则求得的特征向量$\boldsymbol{\omega}$即为三个指标的权重。根据权重，我们可求出当前 CNC 执行第一道工序和第二道工序的得分情况：

$$\mathrm{score}=\begin{bmatrix}\mathrm{a}_1 & 0 & \mathrm{a}_3\\ 0 & \mathrm{a}_2 & \mathrm{a}_3\end{bmatrix}\cdot\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\end{pmatrix}$$

然后将score矩阵进行归一化，得到两个相加为 1 的分数 score1 与 score2，分别代表当前 CNC 进行第一道工序和第二道工序的得分情况。若 score1 较大则当前 CNC 负责第一道工序，反之则负责第二道工序，由此可以确定在处理两道工序物料的时候 CNC 的分配方式。

5.2.2 两道工序模型所使用的二维编码策略

如问题分析中阐述，进行两道工序的操作由于每台 CNC 只能进行一道工序，可以进行第一道和第二道工序的 CNC 是事先确定好的。这样一道工序模型的一维编码就不能用来描述两道工序模型中每个物料的加工顺序，则我们提出新的二维编码策略：

$$f=\begin{bmatrix}\omega_1,\omega_2,\omega_3,\dots,\omega_n\\ \beta_1,\ \beta_2,\beta_3,\dots,\beta_n\end{bmatrix};\quad \omega_i\in M,\beta_i\in N$$

其中集合 M 和 N 分别表示进行第一道与第二道工序的 CNC 编码集合。此种编码方式通过ω,β两个字符分属不同集合的方式来描述负责执行两道工序的 CNC 编号。此编码策略的每一列表示一个物料的加工位置：每一列的第一行元素ω_i表示该物料的第一道工序在编号为ω_i的 CNC 处执行，第二行元素β_i表示该物料的第二

道工序在编号为β_i的 CNC 处执行。这种编码方式可以清晰地表达出两道工序模型中每个物料第一道和第二道工序的加工位置和顺序，从而描述出整个调度流程，能够表达所有可行解。

以上两点为两道工序 RGV 调度模型相对于一道工序的调度模型需要添加和修改的部分。在两道工序的 RGV 调度模型中我们首先要使用层次分析法分析出每个 CNC 负责第几道工序，然后在遗传算法中采用了二维编码的方式编码个体基因型，从而能够描述二维模型的所有可行解。除了这两点之外，两道工序 RGV 的遗传算法模型建立方法与一道工序模型相同。

5.4 可能发生故障情况下的 RGV 调度策略模型

第三种情况要求考虑在有 1%可能发生故障的条件下，分别对一道工序和两道工序的 RGV 调度策略进行建模。在建立模型的前两个部分 5.2 和 5.3 中我们已经建立了加工一道和两道物料的 RGV 调度模型。由于故障是随机产生的，因此我们基于已建立的两个模型动态模拟随机故障产生的过程，由此体现出随机故障对调度效率的影响。

根据应用场景，CNC 在加工过程中有 1%的可能性发生故障，一旦发生故障则需要 10~20 分钟人工排除（在这里我们取 15 分钟），并且正在加工的物料报废。使用动态模型模拟故障产生的过程，首先我们要明确随机故障产生的概率，然后设计模拟随机故障的方法。

5.4.1 确定随机故障产生概率

根据统计，CNC 在加工过程中有 1%的概率发生故障，设 CNC 加工过程中单位时间故障发生的次数为λ_p，由于发生故障是一个随机事件，且在连续时间的每一时刻发生故障的概率都相同，则可以知道故障的产生是一个连续时间的泊松过程，它的概率密度函数为：

$$\mathrm{P(k,\tau)} = \frac{\left(\lambda_p \tau\right)^k e^{-\lambda\tau}}{k!},\ k = 0,1,2,\ldots..$$

其中 k 为故障发生的次数，τ表示一段连续的时间。根据泊松分布的概率密度函数可以求出对于一段连续时间 t 内故障发生次数的期望是：

$$\mathrm{E(k)} = \lambda_p t$$

根据这个原理，我们首先确定λ_p。根据故障发生概率的定义：

$$\text{故障概率} = \frac{\text{故障时间}}{\text{总运行时间}}$$

可以知道，故障概率为 1%，总运行时间为 8 小时（一个班次），则有 288 秒的时间是故障时间。由于每次故障排除时间我们取为 15 分钟，也就是 900 秒，所以换算成次数也就是每个班次平均发生$\frac{288}{900} = 0.32$次故障。由此可以知道，若 8 小时（28800 秒）平均发生 0.32 次故障，则平均每秒发生故障次数为：

$$\lambda_p = \frac{0.32}{28800} = \frac{0.01}{900} = 0.000111\ldots$$

至次我们确定了一段时间 t 内 CNC 发生故障的概率$\lambda_p t$。

5.4.2 模拟随机故障的方法

确定一段时间内 CNC 发生故障的概率，则使用如下的方法模拟随机故障的产生，建立考虑故障的调度策略模型：

（1）根据一道工序或两道工序首先建立遗传算法模型；

（2）对每一代的个体计算该序列所描述调度过程所用的时间：首先根据该序列的顺序从头开始计算每一个步骤所花的时间，并把分步时间累加。若累计时间达到八小时，立即停止模拟；

（3）每一次 CNC 加工物料时，产生均匀随机数判断 CNC 是否会出现故障：

在模拟过程中已知 CNC 进行某一道工序的时间长度 t，那么可以知道在这段加工时间内，CNC 的故障概率为：$\lambda_p t$. 这时产生一个 0-1 均匀分布的随机数，若随机数落在区间$[0, \lambda_p \mathrm{t}]$则认为该 CNC 产生故障。

（4）对于产生故障的 CNC，其正在加工的物料报废，即下一次其发出空闲信号的时间由加工时间变为人工排除故障的 15 分钟。故障排除后 CNC 重新发出空闲信号，再次加入工作队列中。

以上部分建立了动态模拟随机故障产生的模型，这个动态模型是在不考虑故障的一道或两道工序 RGV 调度策略模型的基础上建立的。具体的方法首先是求出平均每秒钟平均故障发生次数，然后求出每段加工时间 CNC 发生故障的概率，再使用随机数模拟的方式判断该 CNC 是否故障。这样在每次使用 CNC 加工时都判断是否故障，就能在原来不考虑故障模型的基础上得到一道和两道工序的 RGV 调度模型。

六. RGV 调度模型的改进

针对已经建立的基础遗传算法模型，我们提出了三点改进方法：多目标优化的改进，针对“早熟收敛”问题的改进，和对初始种群产生方法的改进。

6.1 多目标优化的改进

在原始调度模型中我们采用的适应性函数为：

$$Fitness = num - p$$

此目标函数的含义是固定时间段（一个班次）内加工的物料数。这个目标函数定义的进化方向为增加固定时间内的产出，以此提高效率。另一种提高效率的方法是从成本层面考虑：CNC 加工系统的首要任务是加工物料。如果能够使尽可能多的电能用于加工而不是移动，也可以提高系统的效率。基于这个目标我们设计新的适应性函数：

$$Fitness = w_1(num - p) - w_2 \sum_i Ei$$

其中 E 表示调度过程中移动的距离，其求和为整个工作班次中移动的总距离。由于我们的目标是使移动总距离最小从而减少移动消耗电能来提高效率，因此这一项为负。其中 w_1, w_2 为正系数。

6.2 针对“早熟收敛”现象的改进

在实际模型求解过程中发现每次模拟的结果不一定会收敛到最优值和次优值，在有的模拟中结果会收敛到一个较小的值并且无法跳出。这种现象被称为“早熟”现象。虽然这种情况很快收敛但结果被困在局部最优解，无法在全局范围内搜索。为了避免

早熟收敛现象，我们在交叉算法中结合模拟退火算法，保证了全局寻优，避免早熟现象。模拟退火算法能够保证全局寻优的原因是这种算法采用了 Metropolis 准则[3], 也被称为接受准则，接受概率计算公式为：

$$P_{i,j}=\begin{cases}1, & E_i \geq E_j \\ e^{-\frac{\Delta E_{i,j}}{t}}, & E_i < E_j\end{cases}$$

设当前状态为 i, 该状态的能量为 E_i, j 是状态 i 邻域内的一个新状态，它的能量为 E_j。如果当前状态 i 的能量较低，则接受当前状态 i 作为新状态；如果邻域状态 j 的能量比较低，则产生一个随机数，如果随机数低于$e^{-\frac{\Delta E_{i,j}}{t}}$则接受邻域状态 j 为新状态。就这样重复以上的步骤直到系统逐渐趋于能量较低的稳定状态。此改进模型采用模拟退火-交叉算法[7]进行交叉操作，具体步骤为：

（1）设定初温度函数为$\mathrm{t}_0 = k\Delta_o$，其中 k 为常数，选取 k 时使 k 充分大。Δ_0 表示当前种群中适应度函数的极差，也就是$\Delta_0 = \max(Fitness) - \min(Fitness)$。

（2）在精英保留策略和轮盘赌策略选出的 0.5M 个体中选择一对亲本进行交叉；

（3）判断温度是否为 0，若温度为 0 则停止进化；

（4）对亲本进行单点交叉产生两个子代，记亲本分别为 parent1, parent2, 子代分别为 decendent1, decendent2;

（5）计算亲本和子代的适应度函数值，根据 Metropolis 准则判定 decendent1 是否代替 parent1, decendent2 是否代替 parent2;

（6）计算退火函数$\mathrm{t_k} = \theta \cdot t_{k-1}, 0 < \theta < 1$.计算退火函数之后转到第三步迭代。

以上是模拟退火在交叉方法中的应用。此方法结合基础模型中的轮盘赌与精英保留策略可以在加快收敛速度的同时保证在全局范围内搜索最优解。

6.3 初始种群产生方法的改进

原模型中初始种群的产生方法为随机数产生法。但从实际求解情况观察，得到的收敛结果并不十分理想。在对结果的分析观察中发现，序列中部分编码呈周期性的结果适应度函数值较高。因此，我们将初始种群的编码方式改进为周期性编码，方式如下：

对于一道工序的模型，仍然采用一维编码，但是初始序列使用周期性编码：

$$f = T, T, T, \ldots . T_m$$

$$\mathrm{T} = \omega_1, \omega_2, \ldots . , \omega_8, \ \ \omega_\mathrm{i} \in \{1,2, \ldots ,8\}$$

对于两道工序的模型，仍然采用二维编码，但是初始序列使用周期性编码：

$$f = T, T, T, \ldots . T_m$$

$$\mathrm{T} = \begin{cases}\omega_1, \omega_2, \omega_3, \ldots , \omega_8 \\ \gamma_1, \gamma_2, \gamma_3, \ldots , \gamma_8\end{cases}, \qquad \omega_i \in M, \gamma_i \in N$$

其中集合 M 和 N 分别表示进行第一道与第二道工序的 CNC 编码集合。

七. 改进 RGV 调度模型在三组参数下的求解

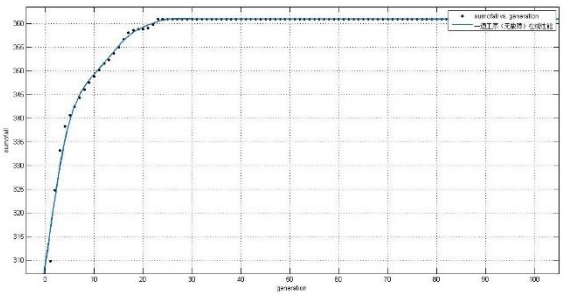
7.1 一道工序且不考虑发生故障模型求解

根据改进的遗传算法模型，采用一维周期编码方式产生初始种群，采用精英保留策略、轮盘赌选择方法、结合模拟退火的单点交叉方法，分别用三组参数求解调度模型，最终求出的最优解如下表所示：

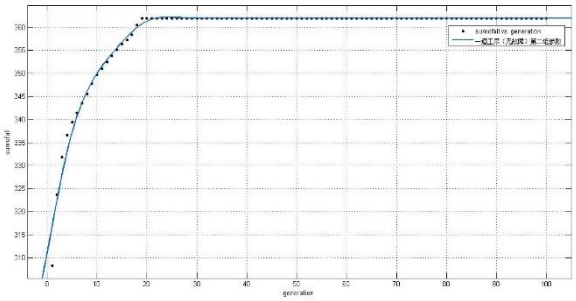
参数组编号	8 小时内加工物料数收敛结果
1	361
2	345
3	373

表 3 第一种情况最优解

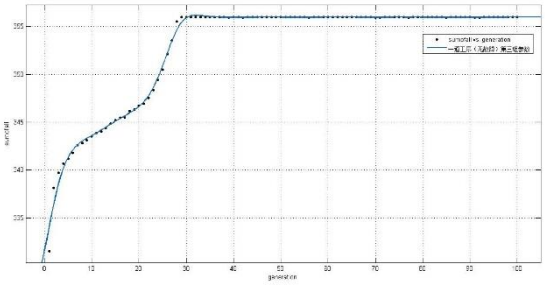
改进的遗传算法模型分别用三组参数求解，下面将每一组的进化过程使用适应度-进化代数图像表示，如下三张图所示。



第一组参数



第二组参数



第三组参数

图 4 一道工序不考虑故障模型收敛情况

以上三图表示三组参数下一道工序不考虑故障的遗传算法模型进化过程。算法收敛之后得到最优解的基因编码即为 RGA 调度策略。因篇幅限制，调度过程详细记录在了附件表格中。

7.2 两道工序且不考虑发生故障的模型求解

7.2.1 层次分析法求解结果

模型使用层次分析法判断 8 个 CNC 中每一个负责第几道工序，过程分别为构建判断矩阵，求矩阵特征向量与特征值，进行一致性检验，通过得分计算出负责第一道和第二道工序的 CNC。

（1）三组参数分别得到的一致性检验结果如下表所示：

CNC#1

参数组编号	权重向量	一致性比率指标 CR
1	[0.0633, 0.0737, 0.9953]	7.6567e-16
2	[0.0968, 0.0596, 0.9935]	3.8284e-16
3	[0.0533, 0.1465, 0.9878]	0

表 4 层次分析法一致性检验结果

前表给出了一号 CNC 通过层次分析法得到的权重向量和一致性比率指标，可以看到一号 CNC 的层次分析法通过了一致性比率检验（CR＜0.1）。完整层次分析法一致性检验结果表请见附件。

（2）层次分析法结果

根据层次分析法得到在三组参数时 CNC 分别应该如何分配，结果如下表：

参数组编号	负责第一道工序 CNC	负责第二道工序 CNC
1	2，5，6，7	1，3，4，8
2	1，5，7	2，3，4，6，8
3	2，3，5	1，4，6，7，8

表 5 CNC 两道工序分配结果

表中结果即为两道工序模型分配工序的策略。根据此结果产生遗传算法的初始种群，用遗传算法求解。

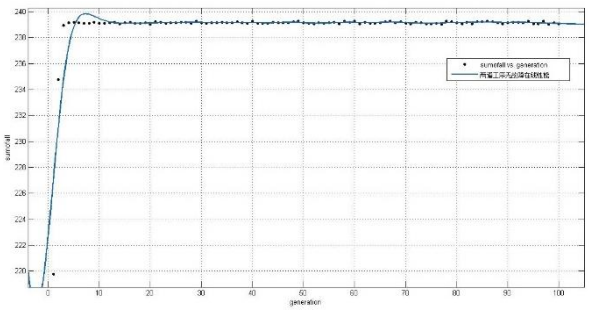
7.2.2 遗传算法求解结果

根据层次分析法结果分配 CNC 工序后，使用改进后的不考虑故障的两道工序模型求解算法，找到最优 RGA 调度方案。求解得到收敛结果如下：

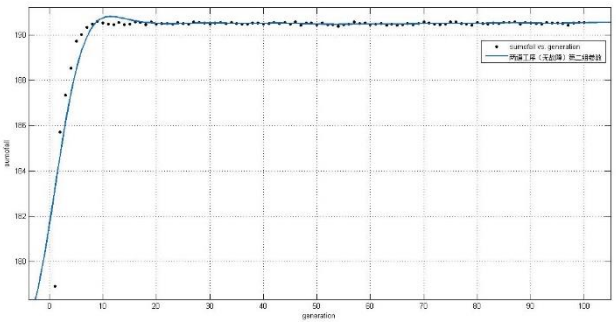
参数组编号	8 小时内加工物料数收敛结果
1	240
2	195
3	189

表 6 两道工序不考虑故障最优解

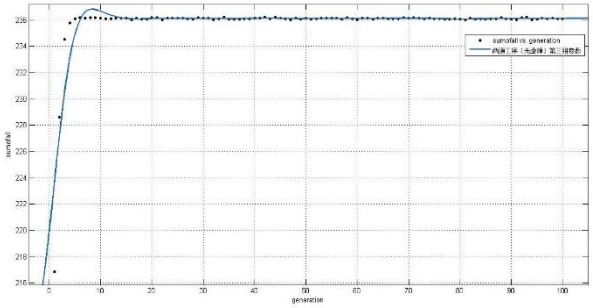
遗传算法进化过程如图所示：



第一组参数



第二组参数



第三组参数

图 5 两道工序不考虑故障遗传算法收敛过程

由图中可以看到在两道工序的情况下算法使用三组参数都可以正常收敛，并且可以看

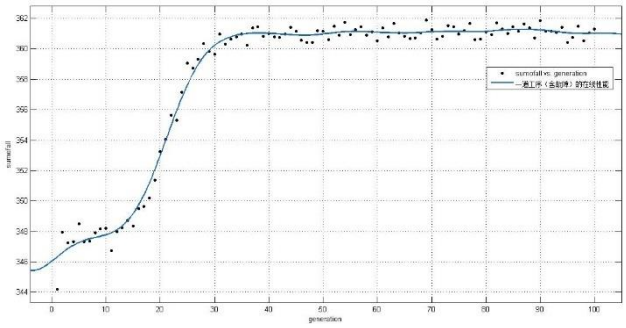
出都可以在十代左右收敛到最优解附近，这说明了修改的遗传算法模型的设计与实现正确，能够通过进化找到最优解。在每一种参数下最优解对应的调度策略详细记录在了附件表格中。

7.3 一道工序且可能发生故障模型求解

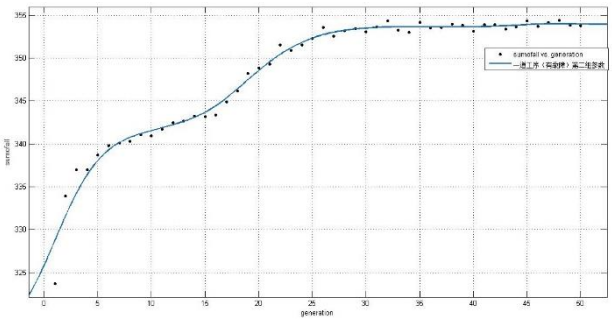
第三种情况相对于第一道工序在遗传算法的建立和求解部分没有改变，而是加入了动态模拟的故障事件。在 8 小时内模型模拟了实际生产系统的运转和故障，并且在此基础上使用遗传算法找到最优的调度方案。下表反映的是在只进行一道工序并且可能发生故障的最优解（能够加工的最大物料数）。

参数组编号	8 小时加工物料数	发生故障数
1	342	2
2	330	3
3	341	3

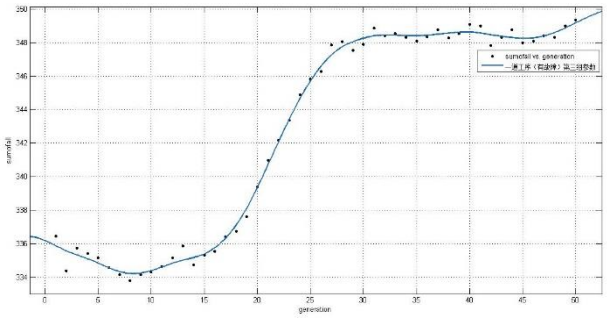
表 7 一道工序考虑故障最优解



第一组参数



第二组参数



第三组参数

图 6 一道工序考虑故障遗传算法收敛过程

上图分别展示了改进后一道工序考虑故障模型的收敛过程。由图像可知在加入随机模拟故障的情况下算法仍可收敛，但收敛所需要的遗传代数明显变长；对比不考虑故障的状态散点相对于拟合线更加分散，并且能够达到的最大加工物料数量有所下降。考虑故障的情况更加符合实际情况，求解验证了模型的正确性。

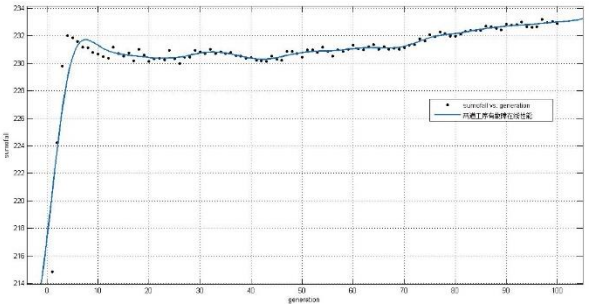
7.4 两道工序且可能发生故障模型求解

两道工序且考虑故障的模型结合了情况二和情况三的求解方法。此模型中运用的编码方式、层次分析法分配 CNC 处理工序与情况二完全相同；在故障模拟方面使用与情况三相同的动态随机故障模拟方法，在遗传算法的基础上模拟实际情况并寻找最优

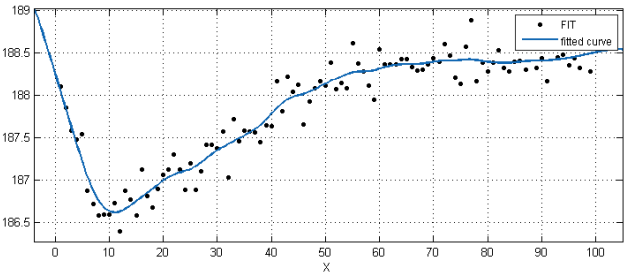
调配方案。下表显示的是在此情况下带入三组参数，分别得到的最大加工物料数和发生的故障数。

参数组编号	8 小时加工物料数	发生故障数
1	231	2
2	186	2
3	182	2

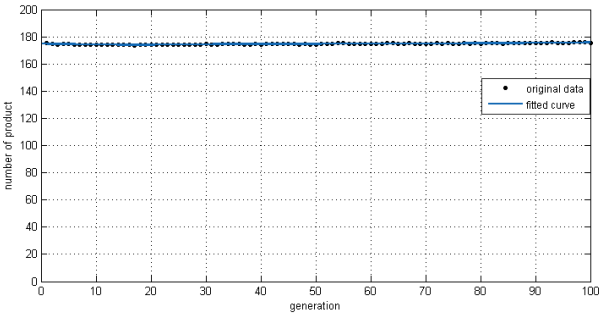
表 8 两道工序可能发生故障模型最优解



第一组参数



第二组参数



第五组参数

图 7 两道工序考虑故障模型收敛过程

上图表示两道工序并考虑故障发生条件下改进遗传算法的收敛过程。与情况三相同，加入随机故障模拟之后最大加工物料数有所下降，并且收敛速度变慢、散点分散度较大。这种情况更符合实际，在可能故障的条件下验证了模型的正确性。另外，由于每台 CNC 只操作一个工序，因此总的调度组合数变少，可以看到模型迅速收敛。同样地，此最优解对应的详细调度过程在附件表格中。

八. RGV 调度模型实用性与有效性评估

8.1 遗传算法模型算法性能分析

遗传算法的性能分析主要通过在线性能和离线性能评估。其中在线性能主要评价遗传算法的收敛性，离线性能评价算法的动态性。在这里我们主要关注遗传算法的收敛情况，所以选取在线性能作为评价指标。在线性能定义为：

$$\mathrm{X(s,t)} = \frac{1}{\mathrm{T}}\sum_{t=1}^{T} F(t)$$

其中 X(s,t)为遗传算法在策略 s 下的第 T 代在线性能，F(t)为第 t 代适应函数的平均值。下图分别是原始模型和改进模型的在线性能曲线。

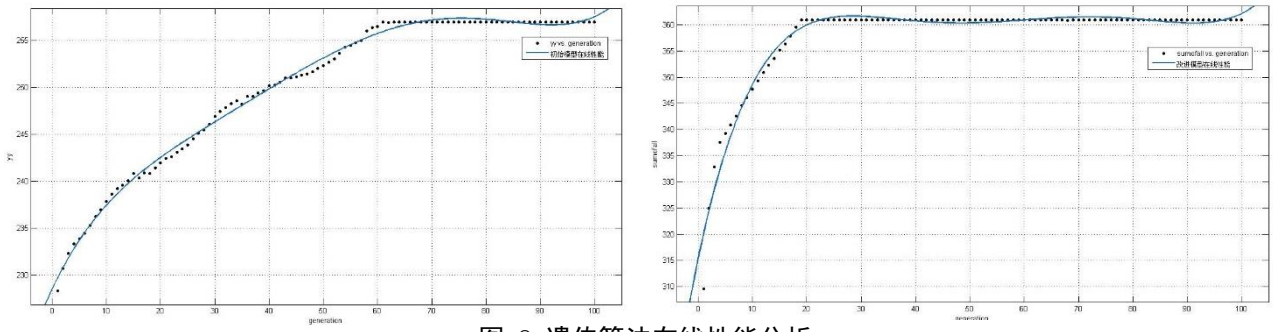


图 8 遗传算法在线性能分析

通过曲线我们可以得出两点结论：

（1）原始模型与改进模型每一代的适应度均值都能有效收敛于某一常数，说明此算法具有较好的在线性能。

（2）从收敛速度来看可以明显观察改进模型的收敛速度更快，并且收敛到了更高的数值。这说明模型的改进不但加快了收敛并且保证了全局最优，这说明模型的改进是有效的。

8.2 模型敏感度分析

敏感度分析是观察模型对一个重要参数的改变具有什么样的变化。在这里我们对改进后的模型对于变异率的敏感度进行分析。在其他参数保持不变的情况下，逐渐改变变异率，得到每次的函数图像如下：

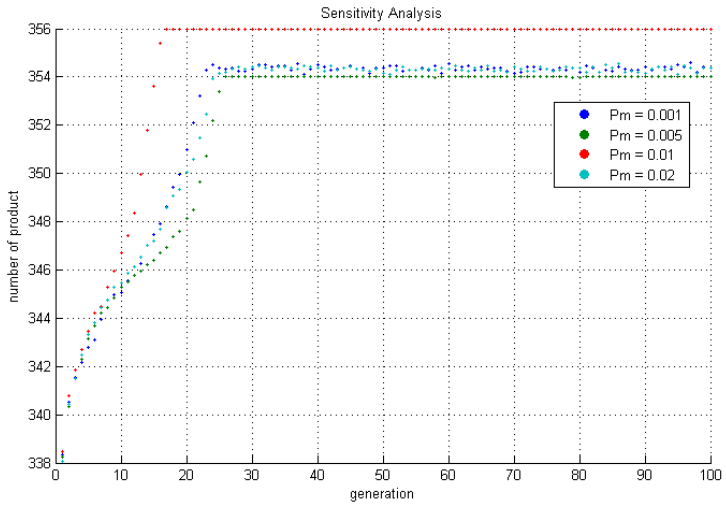


图 9 模型敏感度分析

分别计算变异率改变对最优解变化的影响，从图中可以看到当 Pm 变大 5 倍、10 倍和 20 倍，模型都可以收敛，而且收敛结果受到的影响都在 0.6%以内。因此说明了模型的稳定性。

8.3 使用 Tecnomatix Plant Simulation 进行算法模拟

为了验证模型的实用性，我们使用 Tecnomatix Plant Simulation 仿真软件仿真了考虑发生故障的情况下一道工序物料的加工过程。仿真中使用了参数相同的遗传算法，证明了模型的实用性。仿真截图如下：

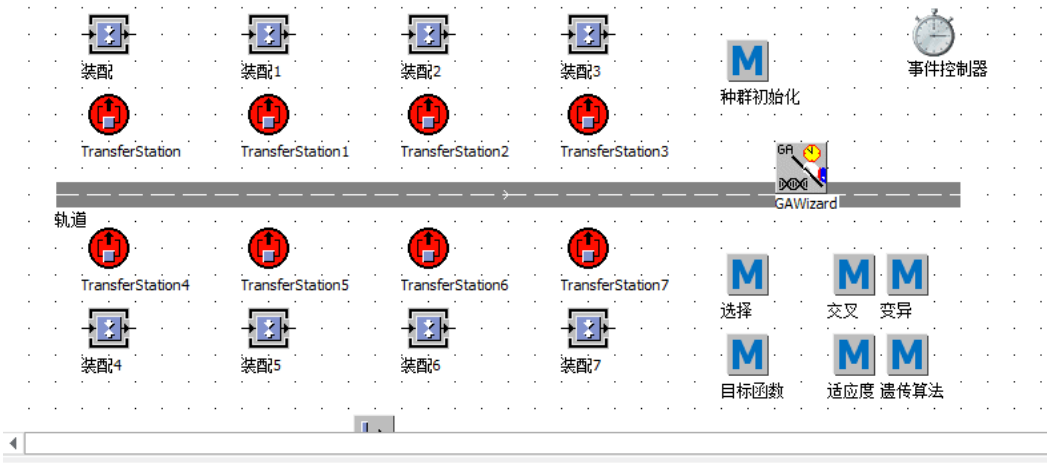
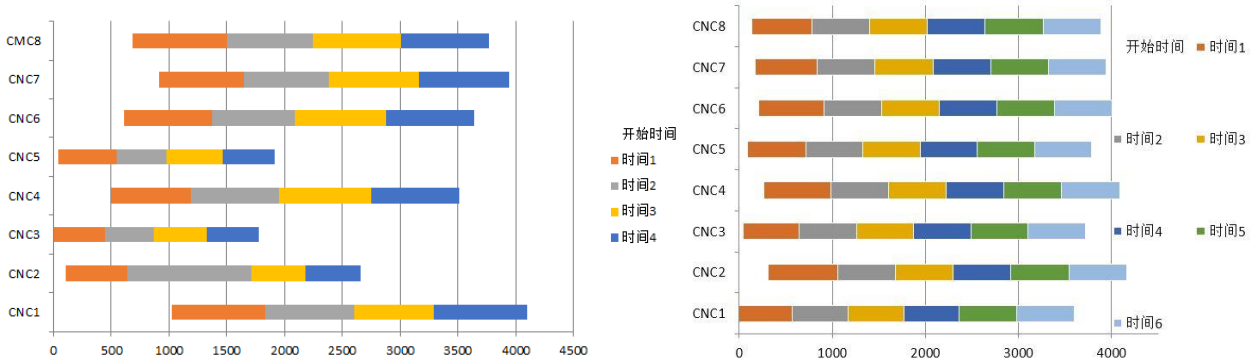


图 10 Tecnomatix Plant Simulation 算法模拟

8.4 使用甘特图表示的调度策略



两道工序调度策略　　一道工序调度策略

图 11 使用甘特图表达的调度策略

上图是甘特图表达的调度策略。甘特图是用来直接反映和指导工程项目规划的图表。左图两道工序的调度策略是考虑故障的模型求解得到的最优解（使用第三组参数），右图是考虑故障的一道工序模型求解得到的最优解（使用第三组参数）。为了表达方便，我们在图中选取了每台 CNC 处理的前四个物料。由此，甘特图反映了模型求解出的最佳调度策略。因此，可知模型的实际应用价值较强，实用性较高。

九. 模型评价

6.1 模型的优缺点

模型优点：

1. 遗传算法相比朴素搜索模型时间复杂度大幅减少，使搜索寻优效率大大提高；
2. 在遗传算法中结合多种选择、交叉策略，既加快了模型收敛，又保证了算法全局寻优，因此改进后的遗传算法模型效率更高，寻优能力更强；
3. 在解决故障问题时，基于泊松过程原理采用了动态随机模拟的方法，能精确地描述故障发生的可能性，而且更接近真实系统运作情况；
4. 在解决两道工序问题时，本模型综合考虑上下料与不同工序的时间成本，用层次分析法分别给出了 3 组参数下的 CNC 工序分配方案，相比随机划分不但提高了模型效率，而且充分利用了时间信息，增强了模型的准确性；
5. 本模型在采用遗传算法优化 RGV 分配序列的时候，通过循环全排列的方式优选初始种群，使得遗传算法能在较短的时间内收敛；
6. 改进模型中引入多目标优化，将系统运行效率纳入指标，实现成本最优化；
7. 使用多种方法检测模型的有效性和实用性，更加具有说服力。

模型缺点：

1. 本文假设条件之一是在加工过程中只有前一道工序完成才会进行下一道，这个条件实际限制了多任务流水线式并行的工作方式，在一定程度上限制了效率；
2. 改进后的初始种群虽然取得了较好的寻优效果，但是在一定程度上削弱了基因的随机性和多样性；
3. 在线性能作为评估指标不能够非常好地衡量算法效率，因为在线性能是基于一次运行得出的评价方法，但遗传算法是基于概率的搜索算法，所以指标应该反映多次算法运行效果。

6.2 模型的改进方向

1. 模型可通过一定的编码规则将 RGV 对具体情况的判优策略加入到染色体中，例如 RGV 能综合考虑分配序列当前位置后的数个分配位置，以决定是否暂时跳过某一分配位置；
2. 在对个体适应度函数的设计中，可通过一定的比例将 CNC 总闲置时间纳入到适应度函数的设计中，以进一步提高 RGV 分配序列的有效性。

十. 参考文献

[1] 程翔宇. 基于遗传算法的多目标动态作业车间调度及应用研究[D].合肥工业大学,2006.

[2] Davis L. Job shop scheduling with genetic algorithms[C]//Proceedings of an international conference on genetic algorithms and their applications. Carnegie-Mellon University Pittsburgh, Pennsylvania, 1985, 140.

[3] Van Laarhoven P J M, Aarts E H L. Simulated annealing[M]//Simulated annealing:

Theory and applications. Springer, Dordrecht, 1987: 7-15.
[4] 王家海,马云雷,肖睿恒.在 eM-Plant 环境下基于遗传算法的 JSP 仿真优化[J].制造业自动化,2011,33(05):6-7+12.
[5] 刘红军,赵帅.一种基于混合遗传算法的车间生产调度的研究[J].制造业自动化,2011,33(17):33-35+59.
[6] 连坤雷,张超勇,高亮,张朝阳.一种基于改进遗传算法的多目标动态调度优化[J].中国制造业信息化,2010,39(03):13-17+21.
[7] 贾彦博,孙立镌.一种改进的遗传模拟退火算法[J].黑龙江科技信息,2007(04):43.
[8] 张海波,贾亚洲,周广文.数控系统故障间隔时间分布模型的研究[J].哈尔滨工业大学学报,2005(02):198-200.
[9] 郑晓伟. 基于遗传算法的作业车间动态调度研究[D].武汉理工大学,2008.
[10] 程翔宇. 基于遗传算法的多目标动态作业车间调度及应用研究[D].合肥工业大学,2006.
[11] 刘永强. 基于遗传算法的 RGV 动态调度研究[D].合肥工业大学,2012.
[12] 翟所霞. 基于改进遗传算法的作业车间调度优化方法研究[D].浙江理工大学,2015.
[13] 周晏明. 基于改进遗传算法的多目标作业车间调度研究[D].东北林业大学,2013.
[14] 潘颖,周柏城.基于改进遗传算法的多目标柔性车间调度问题研究[J].装备制造技术,2016(03):266-267+269.
[15] 马秀明. 基于改进遗传算法的车间调度优化及其仿真[D].大连理工大学,2008.

十一. 附件

一、一致性检验完整表：

CNC#1

参数组编号	权重向量	一致性比率指标 CR
1	[0.0633, 0.0737, 0.9953]	7.6567e-16
2	[0.0968, 0.0596, 0.9935]	3.8284e-16
3	[0.0533, 0.1465, 0.9878]	0

CNC#2

参数组编号	权重向量	一致性比率指标 CR
1	[0.0637, 0.0816, 0.9946]	3.8284e-16
2	[0.1025, 0.0695, 0.9923]	0
3	[0.0572, 0.1729, 0.9833]	-3.8284e-16

CNC#3

参数组编号	权重向量	一致性比率指标 CR
1	[0.0524, 0.0738, 0.9959]	0
2	[0.0801, 0.0597, 0.9950]	7.6567e-16
3	[0.0441, 0.1466, 0.9882]	0

CNC#4

参数组编号	权重向量	一致性比率指标 CR
1	[0.0527, 0.0816, 0.9953]	0
2	[0.0801, 0.0597, 0.9950]	7.6567e-16

3		
CNC#5		
参数组编号	权重向量	一致性比率指标 CR
1	[0.0433,0.0738,0.9963]	0
2	[0.0849, 0.0696, 0.9940]	0
3	[0.0441, 0.1466, 0.9882]	7.6567e-16
CNC#6		
参数组编号	权重向量	一致性比率指标 CR
1	[0.0436, 0.0817, 0.9957]	0
2	[0.0663, 0.0598, 0.9960]	0
3	[0.0473, 0.1730, 0.9838]	7.6567e-16
CNC#7		
参数组编号	权重向量	一致性比率指标 CR
1	[0.0358, 0.0738, 0.9966]	0
2	[0.0702, 0.0697, 0.9951]	7.6567e-16
3	[0.0364, 0.1466, 0.9885]	3.8284e-16
CNC#8		
参数组编号	权重向量	一致性比率指标 CR
1	[0.0360, 0.0817, 0.9960]	0
2	[0.0548, 0.0598, 0.9967]	3.8284e-16
3	[0.0323, 0.1731, 0.9844]	7.6567e-16

二、层次分析法程序

MATLAB 程序，调用时输入参数 N，N = 1，2，3

例如：[A,B] = AHP(2)

```
function [first_step_num, m, sec_step_num,n] = AHP(N)
% output:
% first_step_num = a vector of the number of CNC that perform 1st step
% m = how many CNC of 1st step
% sec_step_num = a vector of the number of CNC that perform 2nd step
% n = how many CNC of 2nd step
% input:
% n = 1, 2 or 3, specifies which set of parameters you would like to use
first_step_num = zeros(1,4);
sec_step_num = zeros(1,4);
m = 0; n = 0;
if N == 1
    a1 = 400;  % time for 1st step
    a2 = 378;  % time for 2nd step
    odd = 28;
    even = 31;
elseif N == 2
    a1 = 280;
    a2 = 500;
    odd = 30;
    even = 35;
elseif N==3
    a1 = 455;
    a2 = 182;
    odd = 27;
    even = 32;
```

```
else disp('invalid input');
end

for ii = 1:8
    if  mod(ii,2) == 0
        a3 = even;
    else
        a3 = odd;
    end
    % A = [a1/a1, a1/a2, a1/a3; a2/a1, a2/a2, a2/a3; a3/a1, a3/a2,
a3/a3];
    a1 = a1 * 1.1;
    A = [a1/a1, a2/a1, a3/a1; a1/a2, a2/a2, a3/a2; a1/a3, a2/a3, a3/a3];
    [V,lambda] = eig(A);
    CI  = (lambda(1,1) - 3)/(3 - 1);
    CR = CI / 0.58
    w = V(:,1)
    score = [a1,0,a3; 0,a2,a3]*w ./ normest(w);
    score1 = score(1)/sum(score);
    score2 = score(2)/sum(score);
    fprintf('score1 = %10.8f\n',score1);
    fprintf('score2 = %10.8f\n',score2);
    fprintf('score1-score2 = %30.29f\n',score(1)-score(2));
    if score(1) > score(2)
        first_step_num(m+1) = ii;
        m = m + 1;
    else
        sec_step_num(n+1) = ii;
        n = n + 1;
    end
end
end
```

三、模型一：一道工序不考虑故障适应度函数

MATLAB 代码

```
%一道工序调度模型适应度函数
function
[fit,load_time,unload_time,m]=fitness(RGV_position,N,j)

%RGV_position RGV移动序列
%N RGV移动序列长度，即RGV预设移动长度
%j 第j个RGV移动序列，即遗传算法中的一个染色体

curent_time=0;              %curent_time 当前时间
CNC_finish_time=zeros(8);  %各个CNC完成当前工作的预定时间
finished_num=0;             %整个系统工作8小时所生产的成品数
load_time=zeros(1,N);
unload_time=zeros(1,N);
CNC_doing=zeros(1,8);      %当前各个CNC正在加工的物料序号
%第一组
exchang_time_odd=28;
exchang_time_even=31;
clear_time=25;
time_for_one_step=560;
mov_1_step_t=20;
mov_2_step_t=33;
mov_3_step_t=46;
fitn=330;
```

```
%第二组
% exchang_time_odd=30;
% exchang_time_even=35;
% clear_time=30;
% time_for_one_step=580;
% mov_1_step_t=23;
% mov_2_step_t=41;
% mov_3_step_t=59;
% fitn=310;

%第三组
% exchang_time_odd=27;
% exchang_time_even=32;
% clear_time=25;
% time_for_one_step=545;
% mov_1_step_t=18;
% mov_2_step_t=32;
% mov_3_step_t=46;
% fitn=340;
      for i=1:N
          load_time(i)=curent_time;

           if(mod(RGV_position(j,i),2)==0)
              curent_time=curent_time+exchang_time_even;
           else
              curent_time=curent_time+exchang_time_odd;
           end

           if(CNC_doing(RGV_position(j,i))>0)

unload_time(CNC_doing(RGV_position(j,i)))=load_time(i);
               finished_num=finished_num+1;
               curent_time=curent_time+clear_time;
           end


CNC_finish_time(RGV_position(j,i))=curent_time+time_for_one_st
ep;
           CNC_doing(RGV_position(j,i))=i;

           if(i<N)

if(curent_time<CNC_finish_time(RGV_position(j,i+1)))

curent_time=CNC_finish_time(RGV_position(j,i+1));
               end
               if(abs(floor((RGV_position(j,i+1)+1)/2)-
floor((RGV_position(j,i)+1)/2))>2)
                   curent_time=curent_time+mov_3_step_t;
               elseif(abs(floor((RGV_position(j,i+1)+1)/2)-
floor((RGV_position(j,i)+1)/2))>1)
                   curent_time=curent_time+mov_2_step_t;
               elseif(abs(floor((RGV_position(j,i+1)+1)/2)-
floor((RGV_position(j,i)+1)/2))>0)
                    curent_time=curent_time+mov_1_step_t;
```

```
            end
        end
        if(curent_time>28800)              %当时间到达8小时，返
回适应度
            %finished(j)=1;
            %fprintf('M:%d  num:%d\n',j,finished_num(j));
            fit=finished_num-fitn;
            m=i;
            break;
        end
    end
end
```

四、模型三：一道工序考虑故障的适应度函数

MATLAB 代码

```
%一道工序带故障调度模型适应度函数
function [fit break_num break_time break_fix_time break_CNC
break_product load_time
unload_time]=fitness_breakdown(RGV_position,N,j)
%RGV_position RGV移动序列
%N RGV移动序列长度，即RGV预设移动长度
%j 第j个RGV移动序列，即遗传算法中的一个染色体
break_time=zeros(1,20);   %故障发生时间
break_fix_time=zeros(1,20); %故障结束时间
break_CNC=zeros(1,20);     %故障CNC编号
break_product=zeros(1,20); %故障时物料序号
CNC_doing=zeros(1,8);     %当前各个CNC正在加工的物料序号
load_time=zeros(1,N);
unload_time=zeros(1,N);
CNC_finish_time=zeros(8);  %各个CNC完成当前工作的预定时间

break_num=0;              %发生故障的次数
curent_time=0;            %curent_time 当前时间
finished_num=0;           %整个系统工作8小时所生产的成品数
break_rate=0.01/900;          %CNC一秒钟发生故障的概率
fix_time=15*60;           %故障修理时间，单位为秒s

%第一组
% exchang_time_odd=28;
% exchang_time_even=31;
% clear_time=25;
% time_for_one_step=560;
% mov_1_step_t=20;
% mov_2_step_t=33;
% mov_3_step_t=46;
% fitn=320;

%第二组
% exchang_time_odd=30;
% exchang_time_even=35;
% clear_time=30;
% time_for_one_step=580;
```

```
% mov_1_step_t=23;
% mov_2_step_t=41;
% mov_3_step_t=59;
% fitn=300;

%第三组
exchang_time_odd=27;
exchang_time_even=32;
clear_time=25;
time_for_one_step=545;
mov_1_step_t=18;
mov_2_step_t=32;
mov_3_step_t=46;
fitn=330;

      for i=1:N
          load_time(i)=curent_time;

           if(mod(RGV_position(j,i),2)==0)
              %curent_time=curent_time+exchang_time_even;
              time_add(exchang_time_even);
           else
              %curent_time=curent_time+exchang_time_odd;
              time_add(exchang_time_odd);
           end

           if(CNC_doing(RGV_position(j,i))>0)

unload_time(CNC_doing(RGV_position(j,i)))=load_time(i);
               finished_num=finished_num+1;
               %curent_time=curent_time+clear_time;
               time_add(clear_time);
           end


CNC_finish_time(RGV_position(j,i))=curent_time+time_for_one_st
ep;
           CNC_doing(RGV_position(j,i))=i;

           if(i<N)

if(curent_time<CNC_finish_time(RGV_position(j,i+1)))
                  %curent_time=CNC_finish_time(RGV_position(j,i
+1));

time_add(CNC_finish_time(RGV_position(j,i+1))-curent_time);
               end
               if(abs(floor((RGV_position(j,i+1)+1)/2)-
floor((RGV_position(j,i)+1)/2))>2)
                  curent_time=curent_time+mov_3_step_t;
               elseif(abs(floor((RGV_position(j,i+1)+1)/2)-
floor((RGV_position(j,i)+1)/2))>1)
                  curent_time=curent_time+mov_2_step_t;
               elseif(abs(floor((RGV_position(j,i+1)+1)/2)-
floor((RGV_position(j,i)+1)/2))>0)
```

```
                curent_time=curent_time+mov_1_step_t;
            end
        end
        if(curent_time>28800)              %当时间到达8小时，返
回适应度
            %finished(j)=1;
            %fprintf('M:%d  num:%d\n',j,finished_num(j));
            fit=finished_num-fitn;
            m=i;
            break;
        end
    end
  function time_add(num2add)
%       for t=1:num2add
%          curent_time=curent_time+1;
%       end
      curent_time=curent_time+num2add;
      for n=1:8
            break_n=rand();
            if(break_n<break_rate*num2add)
                break_num=break_num+1;
                break_product(break_num)=CNC_doing(n);
                CNC_doing(n)=0;      %第n个CNC发生故障
                break_time(break_num)=curent_time;

break_fix_time(break_num)=curent_time+fix_time;
                break_CNC(break_num)=n;
                CNC_finish_time(n)=curent_time+fix_time;

            end
      end

  end
end
```

五、一道工序调度模型遗传算法求解程序（主程序）

MATLAB

```
%一道工序调度模型遗传算法求解程序
N=400;%染色体长度
M=800; %初始种群大小
%RGV_position=[1,randi(8,1,N-1)];
%RGV_position=random('poisson',4,1,N);
RGV_position = ones(M,N);
RGV_position_son=ones(M/2,N);
RGV_position_selected=ones(M/2,N);
RGV_position_next=ones(M,N);
load_time=0;        %加工CNC上料时间
unload_time=0;      %加工CNC下料开始时间

v=[8 7 6 5 4 3 2 1];
f=perms(v);
L=length(f);
ran=floor(rand(1,M)*L);
rgv=1;
```

```
for j=1:M                  %确定初始种群
    for i = 1:(N/8)
    %rgv= [rgv, randperm(8,8)];
    rgv= [rgv, f(ran(j),:)];
    end
    RGV_position(j,:)=rgv(1:N);
    rgv=1;
end
fit=zeros(1,M);
fit_selected=zeros(1,M/2);
gen=100; %遗传代数

for g=1:gen
    M=size(RGV_position,1);
    m=floor(M/2);
for j=1:M
    fit(j)=fitness(RGV_position,N,j);
end
[finished_sort fit_index]=sort(fit,'descend');
for i=1:m              %淘汰适应度排在后半段的个体
    RGV_position_selected(i,:)=RGV_position(fit_index(i),:);
    fit_selected(i)=fit(fit_index(i));
end
fit_sum=sum(fit_selected);
p=zeros(1,m);           %计算轮盘赌中被选择的概率
ppx=zeros(1,m);
for i=1:m
    p(i)=fit_selected(i)/fit_sum;
    if(i==1)
        ppx(i)=p(i);
    else
        ppx(i)=ppx(i-1)+p(i);
    end
end

for ii=1:2:m          %交叉产生M/2个个体
father=rand();
mother=rand();
    for i=1:m  %根据轮盘赌在剩下的种群中确定父亲和母亲
        if father<=ppx(i)
            SelFather =RGV_position_selected(i,:);
            break
        end

    end
    for i=1:m
        if mother<=ppx(i)
           SelMother =RGV_position_selected(i,:);
           break
        end
    end
posCut=floor(rand()*N)+1;   %随机确定交叉点
lenCut=floor(rand()*N)+1;   %随机确定剪切长度
```

```
if(posCut+lenCut>N)
    lenCut=N-posCut;
end
Cut1=SelFather(posCut:posCut+lenCut);  %交叉染色体
Cut2=SelMother(posCut:posCut+lenCut);

RGV_position_son(ii,:)=SelFather;
RGV_position_son(ii+1,:)=SelMother;

RGV_position_son(ii,posCut:posCut+lenCut)=Cut2;
RGV_position_son(ii+1,posCut:posCut+lenCut)=Cut1;

end
%产生变异
by=[];
while ~length(by)
    by=find(rand(1,M)<0.001);      %产生变异操作的染色体位置
end
B=RGV_position(by,:);           %产生变异操作的初始染色体
for i=1:length(by)
    bw=sort(floor(1+N*rand(1,3)));   %产生变异操作的3个地址
    B(i,:)=B(1,[1:bw(1)-
1,bw(2)+1:bw(3),bw(1):bw(2),bw(3)+1:N]);  %交换位置
end

RGV_position=[RGV_position_selected;RGV_position_son;B];
end

[fit1,load_time,unload_time,m]=fitness(RGV_position,N,1)
```

六、一道工序调度模型考虑故障遗传算法求解程序（主程序）

MATLAB

```
%一道工序带故障调度模型遗传算法求解程序
N=400;
M=800; %种群大小
%RGV_position=[1,randi(8,1,N-1)];
%RGV_position=random('poisson',4,1,N);
RGV_position = ones(M,N);
RGV_position_son=ones(M/2,N);
RGV_position_selected=ones(M/2,N);
RGV_position_next=ones(M,N);

break_happen_time=zeros(M,20);

v=[8 7 6 5 4 3 2 1];
f=perms(v);
L=length(f);
ran=floor(rand(1,M)*L);
rgv=1;
for j=1:M                  %确定初始种群
    for i = 1:(N/8)
    %rgv= [rgv, randperm(8,8)];
    rgv= [rgv, f(ran(j),:)];
```

```
    end
    RGV_position(j,:)=rgv(2:N+1);
    rgv=1;
end

fit_selected=zeros(1,M/2);
gen=50; %遗传代数

for g=1:gen
    g
for j=1:M

[fit(j),break_number(j),break_happen_time(j,:)]=fitness_breakd
own(RGV_position,N,j);
end
[finished_sort, fit_index]=sort(fit,'descend');
for i=1:M/2
    RGV_position_selected(i,:)=RGV_position(fit_index(i),:);
    fit_selected(i)=fit(fit_index(i));
end
fit_sum=sum(fit_selected);
p=zeros(1,M/2);           %计算轮盘赌中被选择的概率
ppx=zeros(1,M/2);
for i=1:M/2
    p(i)=fit_selected(i)/fit_sum;
    if(i==1)
        ppx(i)=p(i);
    else
        ppx(i)=ppx(i-1)+p(i);
    end
end

for ii=1:2:M/2
father=rand();
mother=rand();
    for i=1:M/2  %根据轮盘赌在剩下的种群中确定父亲和母亲
        if father<=ppx(i)
            SelFather =RGV_position_selected(i,:);
            break
        end

    end
    for i=1:M/2
        if mother<=ppx(i)
           SelMother =RGV_position_selected(i,:);
           break
        end
    end
posCut=floor(rand()*N)+1;   %随机确定交叉点
lenCut=floor(rand()*N)+1;   %随机确定剪切长度

if(posCut+lenCut>M/2)
    lenCut=N-posCut;
end
Cut1=SelFather(posCut:posCut+lenCut);
```

```
Cut2=SelMother(posCut:posCut+lenCut);

RGV_position_son(ii,:)=SelFather;
RGV_position_son(ii+1,:)=SelMother;

RGV_position_son(ii,posCut:posCut+lenCut)=Cut2;
RGV_position_son(ii+1,posCut:posCut+lenCut)=Cut1;
end
RGV_position=[RGV_position_selected;RGV_position_son];
end
[fit1,break_num,break_time,break_fix_time,break_CNC,break_prod
uct,load_time,unload_time]=fitness_breakdown(RGV_position,N,1)
;
% for j=1:M
%
fit2(j)=fitness(RGV_position_next,N,j,curent_time,CNC_finish_t
ime,finished_num);
% end
```

六、模型二：两道工序调度模型不考虑故障遗传算法求解程序（主程序）

MATLAB

```
%两道工序调度模型遗传算法求解程序
N=800;
M=800; %种群大小
%RGV_position=[1,randi(8,1,N-1)];
%RGV_position=random('poisson',4,1,N);

%第一组
first_step=[7 6 5 2];       %第一道工序的机器编号
second_step=[8 4 3 1];      %第二道工序的机器编号

%第二组
% first_step=[7 5 1];       %第一道工序的机器编号
% second_step=[8 6 4 3 2];      %第二道工序的机器编号

%第三组
% first_step=[5 3 2];       %第一道工序的机器编号
% second_step=[8 7 6 4 1];      %第二道工序的机器编号


len_fir=length(first_step);  %第一道工序机器数量
len_sec=length(second_step); %第二道工序数量

RGV_position1 = ones(M,N/2);   %第一道工序RGV序列
RGV_position2 = ones(M,N/2);   %第二道工序RGV序列
RGV_position_son1=ones(M/2,N/2);
RGV_position_son2=ones(M/2,N/2);
RGV_position_selected1=ones(M/2,N/2);
RGV_position_selected2=ones(M/2,N/2);
rgv1=[2];
rgv2=[1];

f1=perms(first_step);   %第一道工序编号全排列
```

```
f2=perms(second_step);  %第二道工序编号全排列
L1=length(f1);
L2=length(f2);
ran1=floor(rand(1,M)*L1+1);
ran2=floor(rand(1,M)*L2+1);

for j=1:M                  %确定初始种群
    while(length(rgv1)<N/2+1)
        rgv1= [rgv1, f1(ran1(j),:)];
    end
        RGV_position1(j,:)=rgv1(2:N/2+1);

    while(length(rgv2)<N/2+1)
        rgv2= [rgv2, f2(ran2(j),:)];
    end
        RGV_position2(j,:)=rgv2(2:N/2+1);

    rgv1=[2];
    rgv2=[1];
end

fit=zeros(1,M);              %适应度矩阵
fit_selected=zeros(1,M/2);
gen=100; %遗传代数

for g=1:gen
    M=size(RGV_position1,1);
    m=floor(M/2);
for j=1:M
    fit(j)=fitness_2steps(RGV_position1,RGV_position2,N,j);
end
[finished_sort fit_index]=sort(fit,'descend');
for i=1:m              %淘汰适应度排在后半段的个体
    RGV_position_selected1(i,:)=RGV_position1(fit_index(i),:);
    RGV_position_selected2(i,:)=RGV_position2(fit_index(i),:);
    fit_selected(i)=fit(fit_index(i));
end
fit_sum=sum(fit_selected);
p=zeros(1,m);           %计算轮盘赌中被选择的概率
ppx=zeros(1,m);
for i=1:m
    p(i)=fit_selected(i)/fit_sum;
    if(i==1)
        ppx(i)=p(i);
    else
        ppx(i)=ppx(i-1)+p(i);
    end
end

for ii=1:2:m          %交叉产生M/2个个体
father=rand();
mother=rand();
SelFather=zeros(2,N/2);
SelMother=zeros(2,N/2);
```

```
    for i=1:m  %根据轮盘赌在剩下的种群中确定父亲和母亲
        if father<=ppx(i)
            SelFather(1,:) =RGV_position_selected1(i,:);
            SelFather(2,:) =RGV_position_selected2(i,:);
            break
        end

    end
    for i=1:m
        if mother<=ppx(i)
           SelMother(1,:) =RGV_position_selected1(i,:);
           SelMother(2,:) =RGV_position_selected2(i,:);
           break
        end
    end
posCut=floor(rand()*(N/2))+1;   %随机确定交叉点
lenCut=floor(rand()*(N/2))+1;   %随机确定剪切长度

if(posCut+lenCut>N/2)
    lenCut=N/2-posCut;
end

CutF1=SelFather(1,posCut:posCut+lenCut);  %交叉染色体
CutF2=SelFather(2,posCut:posCut+lenCut);
CutM1=SelMother(1,posCut:posCut+lenCut);
CutM2=SelMother(2,posCut:posCut+lenCut);

RGV_position_son1(ii,:)=SelFather(1,:);
RGV_position_son2(ii,:)=SelFather(2,:);

RGV_position_son1(ii+1,:)=SelMother(1,:);
RGV_position_son2(ii+1,:)=SelMother(2,:);

RGV_position_son1(ii,posCut:posCut+lenCut)=CutM1;
RGV_position_son2(ii,posCut:posCut+lenCut)=CutM2;
RGV_position_son1(ii+1,posCut:posCut+lenCut)=CutF1;
RGV_position_son2(ii+1,posCut:posCut+lenCut)=CutF2;

end

RGV_position1=[RGV_position_selected1;RGV_position_son1];
RGV_position2=[RGV_position_selected2;RGV_position_son2];
end
[fit1,load_time1,load_time2,unload_time1,unload_time2,m]=fitne
ss_2steps(RGV_position1,RGV_position2,N,1);
```

七、模型四：两道工序调度模型考虑故障遗传算法求解程序（主程序）

MATLAB

```
    %两道工序带故障调度模型遗传算法求解程序
    N=800;
    M=800; %种群大小
    %RGV_position=[1,randi(8,1,N-1)];
    %RGV_position=random('poisson',4,1,N);

    %第一组
```

```
% first_step=[7 6 5 2];        %第一道工序的机器编号
% second_step=[8 4 3 1];       %第二道工序的机器编号

%第二组
% first_step=[7 5 1];         %第一道工序的机器编号
% second_step=[8 6 4 3 2];       %第二道工序的机器编号

%第三组
first_step=[5 3 2];         %第一道工序的机器编号
second_step=[8 7 6 4 1];       %第二道工序的机器编号

len_fir=length(first_step);  %第一道工序机器数量
len_sec=length(second_step); %第二道工序数量
RGV_position1 = ones(M,N/2);   %第一道工序RGV序列
RGV_position2 = ones(M,N/2);   %第二道工序RGV序列
RGV_position_son1=ones(M/2,N/2);
RGV_position_son2=ones(M/2,N/2);
RGV_position_selected1=ones(M/2,N/2);
RGV_position_selected2=ones(M/2,N/2);
rgv1=2;
rgv2=1;
f1=perms(first_step);   %第一道工序编号全排列
f2=perms(second_step);  %第二道工序编号全排列
L1=length(f1);
L2=length(f2);
ran1=floor(rand(1,M)*L1+1);
ran2=floor(rand(1,M)*L2+1);

for j=1:M                    %确定初始种群
    while(length(rgv1)<N/2+1)
        rgv1= [rgv1, f1(ran1(j),:)];
    end
        RGV_position1(j,:)=rgv1(2:N/2+1);

    while(length(rgv2)<N/2+1)
        rgv2= [rgv2, f2(ran2(j),:)];
    end
        RGV_position2(j,:)=rgv2(2:N/2+1);

    rgv1=[2];
    rgv2=[1];
end


fit=zeros(1,M);              %适应度矩阵
break_num=zeros(1,M);         %发生故障数
fit_selected=zeros(1,M/2);
gen=50; %遗传代数

for g=1:gen
    g
    M=size(RGV_position1,1);
```

```
       m=floor(M/2);
    for j=1:M

[fit(j),break_num(j)]=fitness_2steps_with_breakdown(RGV_positi
on1,RGV_position2,N,j);
    end
    [finished_sort fit_index]=sort(fit,'descend');
    for i=1:m              %淘汰适应度排在后半段的个体

RGV_position_selected1(i,:)=RGV_position1(fit_index(i),:);

RGV_position_selected2(i,:)=RGV_position2(fit_index(i),:);
       fit_selected(i)=fit(fit_index(i));
    end
    fit_sum=sum(fit_selected);
    p=zeros(1,m);           %计算轮盘赌中被选择的概率
    ppx=zeros(1,m);
    for i=1:m
       p(i)=fit_selected(i)/fit_sum;
       if(i==1)
          ppx(i)=p(i);
       else
          ppx(i)=ppx(i-1)+p(i);
       end
    end

    for ii=1:2:m          %交叉产生M/2个个体
    father=rand();
    mother=rand();
    SelFather=zeros(2,N/2);
    SelMother=zeros(2,N/2);
       for i=1:m  %根据轮盘赌在剩下的种群中确定父亲和母亲
          if father<=ppx(i)
             SelFather(1,:) =RGV_position_selected1(i,:);
             SelFather(2,:) =RGV_position_selected2(i,:);
             break
          end

       end
       for i=1:m
          if mother<=ppx(i)
            SelMother(1,:) =RGV_position_selected1(i,:);
            SelMother(2,:) =RGV_position_selected2(i,:);
            break
          end
       end
    posCut=floor(rand()*(N/2))+1;   %随机确定交叉点
    lenCut=floor(rand()*(N/2))+1;   %随机确定剪切长度
    if(posCut+lenCut>N/2)
       lenCut=N/2-posCut;
    end
    CutF1=SelFather(1,posCut:posCut+lenCut);  %交叉染色体
    CutF2=SelFather(2,posCut:posCut+lenCut);
    CutM1=SelMother(1,posCut:posCut+lenCut);
```

```
    CutM2=SelMother(2,posCut:posCut+lenCut);

    RGV_position_son1(ii,:)=SelFather(1,:);
    RGV_position_son2(ii,:)=SelFather(2,:);

    RGV_position_son1(ii+1,:)=SelMother(1,:);
    RGV_position_son2(ii+1,:)=SelMother(2,:);

    RGV_position_son1(ii,posCut:posCut+lenCut)=CutM1;
    RGV_position_son2(ii,posCut:posCut+lenCut)=CutM2;
    RGV_position_son1(ii+1,posCut:posCut+lenCut)=CutF1;
    RGV_position_son2(ii+1,posCut:posCut+lenCut)=CutF2;
    end
    RGV_position1=[RGV_position_selected1;RGV_position_son1];
    RGV_position2=[RGV_position_selected2;RGV_position_son2];
    end
    [fit1,break_num1,load_time1,load_time2,unload_time1,unload
_time2,break_time,break_fix_time,break_CNC,break_product,m]=fi
tness_2steps_with_breakdown(RGV_position1,RGV_position2,N,1);
```

八、模型三：两道工序调度模型不考虑故障适应度函数（子程序）

MATLAB

```
%两道工序调度模型适应度函数
function
[fit,load_time1,load_time2,unload_time1,unload_time2,m]=fitnes
s_2steps(RGV_position1,RGV_position2,N,j)
N=N/2;
%RGV_position RGV移动序列
%N RGV移动序列长度，即RGV预设移动长度
%j 第j个RGV移动序列，即遗传算法中的一个染色体
curent_time=0;              %curent_time 当前时间
CNC_finish_time=zeros(8);  %各个CNC完成当前工作的预定时间
finished_num=0;             %整个系统工作8小时所生产的成品数
CNC_doing=zeros(1,8);      %当前各个CNC正在加工的物料序号

load_time1=zeros(1,N);
load_time2=zeros(1,N);
unload_time1=zeros(1,N);
unload_time2=zeros(1,N);
%第一组
exchang_time_odd=28;
exchang_time_even=31;
clear_time=25;
time_for_first_step=400;   %加工完成第一道工序时间
time_for_sec_step=378;     %加工完成第二道工序时间
mov_1_step_t=20;           %移动所需的时间
mov_2_step_t=33;
mov_3_step_t=46;
fitn=230;

%第二组
% exchang_time_odd=30;
```

```
% exchang_time_even=35;
% clear_time=30;
% time_for_first_step=280;   %加工完成第一道工序时间
% time_for_sec_step=500;     %加工完成第二道工序时间
% mov_1_step_t=23;
% mov_2_step_t=41;
% mov_3_step_t=59;
% fitn=180;


%第三组
% exchang_time_odd=27;
% exchang_time_even=32;
% clear_time=25;
% time_for_first_step=400;   %加工完成第一道工序时间
% time_for_sec_step=378;     %加工完成第二道工序时间
% mov_1_step_t=18;
% mov_2_step_t=32;
% mov_3_step_t=46;
% fitn=170;

      for i=1:N
          %先进行第一道工序的操作
          first_step_done=0;%标记第一道工序是否生产出产品
          load_time1(i)=curent_time;
           if(mod(RGV_position1(j,i),2)==0)
              curent_time=curent_time+exchang_time_even;   %偶数
机器上下料
           else
              curent_time=curent_time+exchang_time_odd;    %奇数
机器上下料
           end

           if(CNC_doing(RGV_position1(j,i))>0)           %第一道
工序已经生产出产品

unload_time1(CNC_doing(RGV_position1(j,i)))=load_time1(i);
               first_step_done=CNC_doing(RGV_position1(j,i));
           end
           CNC_doing(RGV_position1(j,i))=i;


CNC_finish_time(RGV_position1(j,i))=curent_time+time_for_first
_step;
%            curent_time=curent_time+clear_time;
%            finished_num=finished_num+1;


           if(first_step_done>0)


if(curent_time<CNC_finish_time(RGV_position2(j,i)))
```

```
curent_time=CNC_finish_time(RGV_position2(j,i));   %原地等待第二
道工序的请求
                end
                if(abs(floor((RGV_position2(j,i)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>2)
                    curent_time=curent_time+mov_3_step_t;
                elseif(abs(floor((RGV_position2(j,i)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>1)
                    curent_time=curent_time+mov_2_step_t;
                elseif(abs(floor((RGV_position2(j,i)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>0)
                     curent_time=curent_time+mov_1_step_t;
                end
                %再进行第二道工序的操作
                load_time2(first_step_done)=curent_time;
                if(mod(RGV_position2(j,i),2)==0)
                  curent_time=curent_time+exchang_time_even;   %
偶数机器上下料
                else
                  curent_time=curent_time+exchang_time_odd;    %
奇数机器上下料
                end
                if(CNC_doing(RGV_position2(j,i))>0)

unload_time2(CNC_doing(RGV_position2(j,i)))=load_time2(first_s
tep_done);

curent_time=curent_time+clear_time;            %第二道工序完成，对
熟料进行清洗，成品数加一
                    finished_num=finished_num+1;
                end
                CNC_doing(RGV_position2(j,i))=first_step_done;

CNC_finish_time(RGV_position2(j,i))=curent_time+time_for_sec_s
tep;

                if(i<N)

if(curent_time<CNC_finish_time(RGV_position1(j,i+1)))

curent_time=CNC_finish_time(RGV_position1(j,i+1));   %原地等待下
一个CNC的请求
                    end
                    if(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position2(j,i)+1)/2))>2)
                       curent_time=curent_time+mov_3_step_t;
                    elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position2(j,i)+1)/2))>1)
                       curent_time=curent_time+mov_2_step_t;
                    elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position2(j,i)+1)/2))>0)
                        curent_time=curent_time+mov_1_step_t;
                    end
                end
```

```
            else    %直接移动到下一个CNC转载第一道工序物料

if(curent_time<CNC_finish_time(RGV_position1(j,i+1)))

curent_time=CNC_finish_time(RGV_position1(j,i+1));   %原地等待下
一个CNC的请求
                end
                if(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>2)
                    curent_time=curent_time+mov_3_step_t;
                elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>1)
                    curent_time=curent_time+mov_2_step_t;
                elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>0)
                     curent_time=curent_time+mov_1_step_t;
                end
            end

            if(curent_time>28800)              %当时间到达8小时，返
回适应度
                %finished(j)=1;
                %fprintf('M:%d  num:%d\n',j,finished_num(j));
                fit=finished_num-fitn;
                m=i;
                break;
            end
        end

end
```

九、模型四：两道工序调度模型考虑故障适应度函数（子程序）

MATLAB

```
%两道工序带故障调度模型适应度函数
function
[fit,break_num,load_time1,load_time2,unload_time1,unload_time2
,break_time,break_fix_time,break_CNC,break_product,m]=fitness_
2steps_with_breakdown(RGV_position1,RGV_position2,N,j)
N=N/2;
%RGV_position RGV移动序列
%N RGV移动序列长度，即RGV预设移动长度
%j 第j个RGV移动序列，即遗传算法中的一个染色体

break_time=zeros(1,20);   %故障发生时间
break_fix_time=zeros(1,20); %故障结束时间
break_CNC=zeros(1,20);     %故障CNC编号
break_product=zeros(1,20); %故障时物料序号
CNC_doing=zeros(1,8);     %当前各个CNC正在加工的物料序号
CNC_finish_time=zeros(8);  %各个CNC完成当前工作的预定时间

load_time1=zeros(1,N);
load_time2=zeros(1,N);
unload_time1=zeros(1,N);
```

```
unload_time2=zeros(1,N);

curent_time=0;            %curent_time 当前时间
finished_num=0;           %整个系统工作8小时所生产的成品数
break_num=0;                %8小时发生故障数
break_rate=0.01/900;         %CNC一秒钟发生故障的概率
fix_time=15*60;

%第一组
% exchang_time_odd=28;
% exchang_time_even=31;
% clear_time=25;
% time_for_first_step=400;   %加工完成第一道工序时间
% time_for_sec_step=378;     %加工完成第二道工序时间
% mov_1_step_t=20;          %移动所需的时间
% mov_2_step_t=33;
% mov_3_step_t=46;
% fitn=200;

%第二组
% exchang_time_odd=30;
% exchang_time_even=35;
% clear_time=30;
% time_for_first_step=280;   %加工完成第一道工序时间
% time_for_sec_step=500;     %加工完成第二道工序时间
% mov_1_step_t=23;
% mov_2_step_t=41;
% mov_3_step_t=59;
% fitn=170;


%第三组
exchang_time_odd=27;
exchang_time_even=32;
clear_time=25;
time_for_first_step=400;   %加工完成第一道工序时间
time_for_sec_step=378;     %加工完成第二道工序时间
mov_1_step_t=18;
mov_2_step_t=32;
mov_3_step_t=46;
fitn=170;

      for i=1:N
          %先进行第一道工序的操作
          first_step_done=0;%标记第一道工序是否生产出产品
          load_time1(i)=curent_time;
           if(mod(RGV_position1(j,i),2)==0)
              %curent_time=curent_time+exchang_time_even;   %偶
数机器上下料
              time_add(exchang_time_even);
           else
              %curent_time=curent_time+exchang_time_odd;    %奇
```

```
数机器上下料
             time_add(exchang_time_odd);
          end

          if(CNC_doing(RGV_position1(j,i))>0)          %第一道
工序已经生产出产品

unload_time1(CNC_doing(RGV_position1(j,i)))=load_time1(i);
              first_step_done=CNC_doing(RGV_position1(j,i));
          end
          CNC_doing(RGV_position1(j,i))=i;


CNC_finish_time(RGV_position1(j,i))=curent_time+time_for_first
_step;
%           curent_time=curent_time+clear_time;
%           finished_num=finished_num+1;


          if(first_step_done>0)


if(curent_time<CNC_finish_time(RGV_position2(j,i)))
                  %curent_time=CNC_finish_time(RGV_position2(j,
i));   %原地等待第二道工序的请求
                  time_add(CNC_finish_time(RGV_position2(j,i))-
curent_time);
              end
              if(abs(floor((RGV_position2(j,i)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>2)
                  curent_time=curent_time+mov_3_step_t;
              elseif(abs(floor((RGV_position2(j,i)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>1)
                  curent_time=curent_time+mov_2_step_t;
              elseif(abs(floor((RGV_position2(j,i)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>0)
                   curent_time=curent_time+mov_1_step_t;
              end
              %再进行第二道工序的操作
              load_time2(first_step_done)=curent_time;
              if(mod(RGV_position2(j,i),2)==0)
                 curent_time=curent_time+exchang_time_even;   %
偶数机器上下料
              else
                 curent_time=curent_time+exchang_time_odd;    %
奇数机器上下料
              end
              if(CNC_doing(RGV_position2(j,i))>0)

unload_time2(CNC_doing(RGV_position2(j,i)))=load_time2(first_s
tep_done);
                  %curent_time=curent_time+clear_time;
%第二道工序完成，对熟料进行清洗，成品数加一
                  time_add(clear_time);
```

```
                finished_num=finished_num+1;
            end
            CNC_doing(RGV_position2(j,i))=first_step_done;

CNC_finish_time(RGV_position2(j,i))=curent_time+time_for_sec_s
tep;

            if(i<N)

if(curent_time<CNC_finish_time(RGV_position1(j,i+1)))
                %curent_time=CNC_finish_time(RGV_position1
(j,i+1));   %原地等待下一个CNC的请求

time_add(CNC_finish_time(RGV_position1(j,i+1))-curent_time);
                end
                if(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position2(j,i)+1)/2))>2)
                   curent_time=curent_time+mov_3_step_t;
                elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position2(j,i)+1)/2))>1)
                   curent_time=curent_time+mov_2_step_t;
                elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position2(j,i)+1)/2))>0)
                    curent_time=curent_time+mov_1_step_t;
                end
            end
          else    %直接移动到下一个CNC转载第一道工序物料

if(curent_time<CNC_finish_time(RGV_position1(j,i+1)))
                %curent_time=CNC_finish_time(RGV_position1
(j,i+1));   %原地等待下一个CNC的请求

time_add(CNC_finish_time(RGV_position1(j,i+1))-curent_time);
            end
            if(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>2)
                   curent_time=curent_time+mov_3_step_t;
                elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>1)
                   curent_time=curent_time+mov_2_step_t;
                elseif(abs(floor((RGV_position1(j,i+1)+1)/2)-
floor((RGV_position1(j,i)+1)/2))>0)
                    curent_time=curent_time+mov_1_step_t;
             end
          end

          if(curent_time>28800)             %当时间到达8小时，返
回适应度
            %finished(j)=1;
            %fprintf('M:%d  num:%d\n',j,finished_num(j));
            fit=finished_num-fitn;
            m=i;
            break;
          end
      end
```

```
    function time_add(num2add)
%         for t=1:num2add
%             curent_time=curent_time+1;
%         end
        curent_time=curent_time+num2add;
        for n=1:8
              break_n=rand();
              if(break_n<break_rate*num2add)
                  break_num=break_num+1;
                  break_product(break_num)=CNC_doing(n);
                  CNC_doing(n)=0;       %第n个CNC发生故障
                  break_time(break_num)=curent_time;

break_fix_time(break_num)=curent_time+fix_time;
                  break_CNC(break_num)=n;
                  CNC_finish_time(n)=curent_time+fix_time;

              end
        end

    end
end
```