智能 RGV 的动态调度策略

摘要

本文针对题目所给的智能 RGV 的组成与作业流程，希望能够合理调度轨道式自动引导车，提升作业效率。为此，建立 RGV 动态调度模型。

本题的 RGV 动态调度策略旨在减少 RGV 的运输时间和 CNC 的等待时间，在具体的工作流程中可以转化为工序的排序问题和机器选择问题，将这两个问题的可行解进行编码。工序排序：采用基于工序的实数编码。将总工序数确定为码串总长度，每个基因的值用工件号表示，其中每个工件号出现的次数代表该工件的工序数。如一个可行基因串 1-2-3-1-4-5-6-2, 对应的工作顺序是O_{11}- O_{21}-O_{31}-O_{12}-O_{41}-O_{51}-O_{61}-O_{22}。机器选择排序：采用基于 CNC 分配的实数编码。将总工序定义为编码串，从左至右依次表示工件工序的顺序排列，编码位置代表工序对应的加工机器。

针对一道工序的物料加工情况，每台 CNC 安装同样的刀具，物料可以在任一台 CNC 上加工完成。本问使用改进离散粒子群算法（IDPSO），基于粒子群算法的进化优化机制，引入变异，交叉等操作，对粒子进行离散的位置更新，采用了基于 Intercharge 邻域结构的局部搜索操作。初始解的优劣影响粒子群算法是否能快速收敛和求解质量。我们首先使用考虑工序加工时间和运输时间因子影响的最短工作时间法（short working machine, SWM）,来获得较优的初始解。初始种群的一半采用 SWM 生成，一半用随机化方法生成。多次迭代，计算结果。在一个生产周期内，在第一组系统参数下，CNC 加工完成 377 个物料，实际清洗下线 370；在第二组系统参数下，CNC 加工完成 326 个物料，实际清洗下线 320； 在第三组系统参数下，CNC 加工完成 386 个物料，实际清洗下线 380。在考虑故障率为 1%的情况下，第一组 CNC 加工完成 375 个，实际清洗下线 367 个；第二组 CNC 加工完成 317 个，实际清洗下线 309 个；第三组 CNC 加工完成 362 个，实际清洗下线 356 个。使用 matlab 仿真出甘特图，见附录。

针对两道工序的物料加工情况，这是一个柔性作业车间调度问题（FJSP）。遗传算法是求解此类组合优化问题的有效办法，传统遗传算法有易早熟，容易陷入局部最优的弊端。本文使用改进的遗传算法。将编码作为染色体。在遗传算法的框架下，结合小生境的思想，使用与当前最优解的汉明距离 d 将种群均分为优势子种群和次优子种群，优势子种群为了提高整体的适应度，次优子种群为保持种群的多样性。平衡种群选择压力以克服 GA 容易陷入局部最优的弊端。在子种群中，分别取适应度值前 1%放入精英库，精英直接保留到下一代种群，剩余个体执行交叉，变异。工序染色体使用 IPOX 交叉法作为交叉算子，避免产生不可行的解；机器染色体采用多点交叉（MPX）方式。反复迭代遗传，计算出结果。不考虑故障，一个生产周期内，在第一组系统参数下，生产 253 个；在第二组系统参数下，生产 209 个；在第三组系统参数下，生产 232 个。在存在故障的情况下，第一组系统参数，完成 249 个；第二组系统参数，完成 204 个；在第三组系统参数，完成 228 个。使用 matlab 仿真出甘特图，见附录。

对于模型的改进与完善，提出了多台 RGV 的动态调度模型和求解。在两台 RGV 的情况下，处理问题所给出的情况，使用了粒子群和遗传算法相结合的禁忌搜索算法，让两辆 RGV 小车各自作业，互不干扰，计算出一辆小车在 8 小时内加工完成的物料数。对于情况一的第一组数据，两辆 RGV 小车总共可完成数为 380，相比较单车模型结果，略微提高。对于情况二，两步工序的第一组数据，求解出两辆小车总共可完成数为 276 件，相比较原两工序单 RGV 车模型的结果，有一定的优化。

关键词：作业车间调度，粒子群算法，遗传算法，最短工作时间法，IPOX 交叉法，小生境

一、问题重述及简要分析

1.1 问题重述

如图所示，是一个智能的加工系统，由 8 台数控机床（CNC），一辆自动引导车（RGV），一条直线轨道，一条上料传送带，一条下料传送带组成。RGV 是一种能在 RGV 直线轨道上无人驾驶的小车。它可以根据指令在轨道上任意移动，并自带一个机械手臂，两只机械手爪和物料清洗槽，可以进行上下料和清洗物料等工作。

针对下面三种情况：

1.只有一道工序的物料加工，每台 CNC 安装同样的刀具，物料可以在任意一台上加工。

2.有两道工序的物料加工，每个物料的两道工序分别由不同的 CNC 以此加工。

3.CNC 在加工过程中可能发生故障（概率约为 1%），每次故障排除时间在 10-20 分钟之间，故障排除后即刻可以继续工作，不过物料报废。要求分别考虑一道工序和两道工序的物料加工情况。

完成下列任务：

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。

2.利用表一中的作业参数分别检验模型的实用和算法的有效，给出调度策略和作业效率，并将结果填入表中。

1.2 简要分析

这是一个考虑运输时间的柔性作业车间调度问题（FJSP），FJSP 问题已被证明为 NP-hard 问题。粒子群和遗传算法是求解此类组合优化问题的有效办法。参考杨立熙等人的研究[1],采用一种小生境遗传算法，平衡种群选择压力以克服过早陷入局部最优的缺点，并使用新的初始化方法。

二、假设及关键符号说明

2.1 模型假设与说明

（1）同一台 CNC 同一时刻只能加工同一工件。

（2）所有工序按先后顺序加工，每一道工序加工完成后、运输到下一道加工工序的 CNC。

（3）每个工件加工一旦开始除非 CNC 故障便不得停止。

（4）上料传送带上有源源不断的生料提供且满足 RGV 的工作需求即 RGV 运行至需要作业的 CNC 处时，上料传送带能够将生料送到该 CNC 处。

（5）下料传送带上有足够的空位供机械臂将成料倒下料传送带上送出系统。

2.1 符号说明

J_k	工件集 J = {J_1，J_2, ⋯, J_k, ⋯, J_n}，J_k为第 k 个工件（k = 1,2, ⋯, n）
M_i	机器集 M = {M_1, M_2, ⋯, M_p, ⋯, M_m}, M_p为第 p 台机器（p = 1,2, ⋯, m）
O_{jh}	表示第 j 个工件的第 h 道工序（h = 1,2）
$O_{j(h-1)}$	表示第O_{jh}的前一道工序
$O_{j'h'}$	表示O_{jh}当前所在机器的前一道工序
P_{ijh}	表示第 j 个工件的第 h 道工序在机器 i 上加工所需时间
S_{ijh}	表示第 j 个工件的第 h 道工序在机器 i 上的开始加工时间
C_{ijh}	表示第 j 个工件的第 h 道工序在机器 i 上的结束加工时间
C_i	表示第 j 个工件的完工时间
T_{ie}	表示机器M_i和机器M_e之间的运输时间
T_i	RGV 移动 i 个单位所用的时间（i = 0,1,2,3; $T_0 = 0$）
N_i	RGV 为 CNCi 一次上下料时间
C_{max}	表示最大完工时间

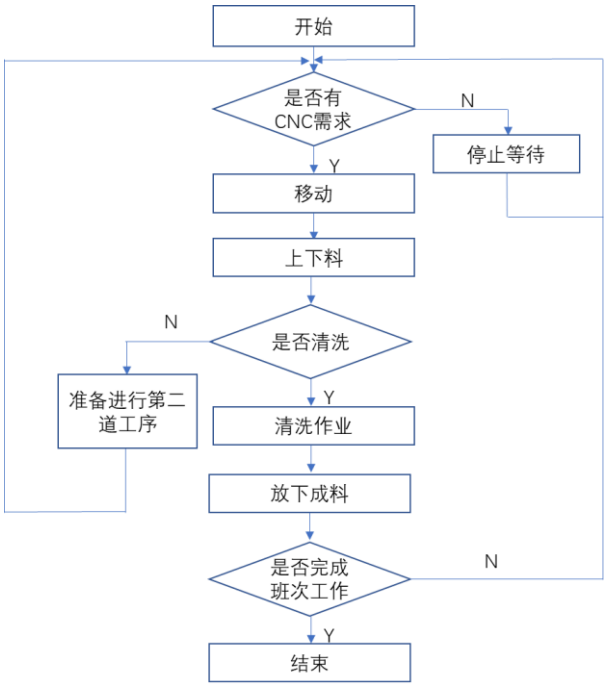
三、问题一

3.1 分析

本题的目标是建立合理的智能 RGV 的动态调度策略以实现在一个工期内生产最多的产品[2]，在每班次连续作业 8 小时的情况下实现效率的最大化。我们将在固定时间中生产更多成料转化为在固定成料要求的情况下所需的生产时间最少，可以转化为最小化最大生产时间的问题。定义系统的作业效率为每小时系统产生的成品个数。

3.2 RGV 工作流程

由于 RGV 同一时间只能执行移动、停止等待、上下料和清洗作业中的一项，所以当 CNC 发出需求时，若 RGV 处于上下料后即将清洗作业时，需要先进行清洗作业以避免任务列表的拥堵。接下来移动到发出需求的 CNC 处进行上下料工作，若之后有清洗作业需要先完成清洗作业，完成后进行等待或者移动。RGV 工作流程图如图一：



图一：RGV 工作流程图

3.3 模型建立的分析

3.3.1 目标函数

首先我们建立目标：最小化最大完工时间：

$$C_{\max} = \min(\max_{1\le j\le n}(C_j)) \tag{3-3-1}$$

3.3.2 约束条件

（1）工序的完成时间应大于等于开始时间与加工时间之和：

$$C_{ijh} \ge P_{ijh} + S_{ijk} \tag{3-3-2}$$

（2）由于一台机器在同一时间只能加工同一工件，所以前后两次工序完成时间只差应大于后一次工序所花费时间：

$$C_{ijh} - C_{ij'h'} \ge P_{ijk} \tag{3-3-3}$$

（3）若工序所在机器的可开始加工时间小于前一道工序运输结束时间，则受运输时间约束；否则，前后两道工序将会在时间上产生冲突：

$$S_{ijh}=\begin{cases}C_{ej(h-1)}+T_{ie} & C_{ij'h'}<C_{ej(h-1)}+T_{ie}\\ C_{ij'h'} & C_{ij'h'}>C_{ej(h-1)}+T_{ie}\end{cases} \tag{3-3-4}$$

（4）两道工序之间的约束，对于一个工件，只有完成了上道工序后才能去完成下一道工序。

$$S_{ihj}\geq C_{i'h'j}+T_{ii'}+N_i$$

(5)p=1,2,⋯,m k=1,2,⋯,n h=1,2

3.3.3 RGV 动态调度最终模型

考虑运输时间和上下料时间的柔性作业车间调度问题的模型:

$$C_{\max}=\min(\max_{1\leq j\leq n}(C_j))$$

$$s.t.\begin{cases}C_{ijh}\geq P_{ijh}+S_{ijk}\\ C_{ijh}-C_{ij'h'}\geq P_{ijk}\\ S_{ijh}=\begin{cases}C_{ej(h-1)}+T_{ie} & C_{ij'h'}<C_{ej(h-1)}+T_{ie}\\ C_{ij'h'} & C_{ij'h'}>C_{ej(h-1)}+T_{ie}\end{cases}\\ S_{ihj}\geq C_{i'h'j}+T_{ii'}+N_i\\ p=1,2,...,m\\ k=1,2,...,n\\ h=1,2\end{cases}$$

3.4 模型求解

3.4.1 最短时间算法

初始解会影响算法的收敛速度和求解的质量。目前使用的较多的方法是随机初始化[3]，这种方法虽然简单但是可能会造成迭代次数的增加。所以在此考虑工序的加工时间和运输时间之和最小的最短工作方法（short working machine，SWM）[4]，赋予各个工序近似最优解的机器来获得初始解，但是为了保持随机性，故初始化采用一半初始解一半随机初始化方法来赋初始解。

SWM 的执行步骤：

（1） 获取工件集总数 J_O_number，每个工件J_i对应的工序数 size(J(i), O);

（2） 设置一个整型数组Id_x,用于记录机器索引号，设置一个整型数组 chrom_M，长度等于总工序数，用于记录 SMW 算法的机器选择部分，初始化数组为 0;

（3） 随机选择一个工件，并选择当前工件的第一道加工工序;

（4） 寻找当前工序的可选择加工机器及加工时间，选择时间最小的那台机器作为当前工序的加工机器，更新Id_x和 chrom_M，并以此作为下一次选择的依据;

（5） 选择当前工件的下一道工序，并记录工序的可选机器及对应加工时间，从Id_x中获取上一道工序加工索引号，计算相邻工序的运输时间，不更新数组;

（6） 将加工时间与运输时间相加，选择相加和最小的那台机器作为当前加工机器，更新Id_x和 chrom_M;

（7） 重复步骤（5)、步骤（6)，直至当前工序的加工机器选择完毕;

（8） 从工件集中选取下一个工件，同时选择工件的第一道工序，重复步骤 (4) ——步骤 (7)，直至工件被选择完毕。

3.4.2 粒子群算法

1.算法原理

粒子群算法[5]是一种基于种群迭代的进化算法，粒子群算法使用无质无量无体积的粒子作为个体，并为每个粒子规定简单的行为规则，从而使的整个粒子群表现出复杂的特性。粒子在搜索空间中根据一定的规则按飞行速度移动，通过其飞行速度和粒子位置的不断飞行搜索，找到算法的最优解。

假设种群规模为 N_p ，求解的搜索空间为 D； $x_i=(x_1,x_2,x_3,...,x_D)$ 表示第 i 个粒子 $(i=1,2,...,N_p)$ 在搜索空间中的位置矢量，表明了当前粒子的优劣； $v_i=(v_1,v_2,...,v_D)$ 表示粒子的飞行速度；根据适应度目标函数计算所有粒子的适应度值，得到每个粒子在搜索空间里的历史最佳位置，就是粒子自生个体的最优位置，由 $pB_i=(pB_{i1},pB_{i2},...,pB_{iD})$ 表示；$gB=(gB_1,gB_2,...,gB_D)$ 表示整个粒子群目前发现的历史最佳位置。

在每一次迭代进化过程中，根据公式 3-4-1 和公式 3-4-2，对粒子的位置和速度进行更新：

$$V_i^{k+1}=\omega V_i^k+c_1r_1(pB_i^k-X_i^k)+c_2r_2(gB^k-X_i^k) \quad (3\text{-}4\text{-}1)$$

$$X_i^{k+1}=X_i^k+V_i^{k+1} \quad (3\text{-}4\text{-}2)$$

其中，ω 为惯性权重因子，通过他可以对粒子的自生速度进行调整，控制收敛性；c_1 和c_2 为加速度系数，表示粒子对个体最优和全局最优的学习； r_1 和r_2 是两个在[0,1]范围内的随机数。而且，在算法迭代中，需要对粒子的速度和位置进行适当的控制，设置粒子的速度和位置区间为 $[v_{\min},v_{\max}]$ 和$[x_{\min},x_{\max}]$ 。

从粒子的更新公式可知，粒子位置的移动受三部分影响：（1）当前粒子速度对下一代粒子位置的影响。(2）粒子自生记忆对下一代粒子位置的影响，体现了粒子对自身的“认知”模式。(3）种群中的全局最优位置对当前粒子的影响，表示群体信息的“社会”模式。

2.算法流程

（1）进行算法的初始化，对种群中所有粒子的位置和速度进行初始化。

（2）根据问题的适应度函数，对每个粒子的适应度进行计算评价。

（3）将种群中每个粒子的适应度值与它自身所经历过的最佳位置 pBi 的适应度值进行比较，如更好，则替换当前的最优。将所有粒子的最优位置与全局最优进行比较，如更好，则替换当前的全局最优。

（4）根据式 3-4-1 和 3-4-2 更新各个粒子的位置和速度。

（5）判断是否满足算法的终止条件，如果是，则输出当前最优解，否则，转向（2）

3.问题编码

本题的 RGV 动态调度策略减少 RGV 运输时间和等待时间在具体的工作流程中可以转化为工序的排序问题和机器选择问题，将这两个问题融合为一条染色体以表示动态调度的一个可行解。具体编码与解码的如下列文字描述以及图 3-4-1 表示：

（1） 工序排序：采用基于工序的实数编码。将总工序数定为基因串长度，每个基因的为值用工件号表示，其中每个工件号出现的次数代表该工件的工序数。如一个可行基因串 1-2-3-1-4-5-6-2，对应的工作顺序是O_{11}-O_{21}-O_{31}-O_{12}-O_{41}-O_{51}-O_{61}-O_{22}。

（2） 机器选择排序：采用基于 CNC 分配的实数编码。将总工序定义为基因串，从左至右依次表示工件工序的顺序排列，基因位置代表工序对应的加工机器。

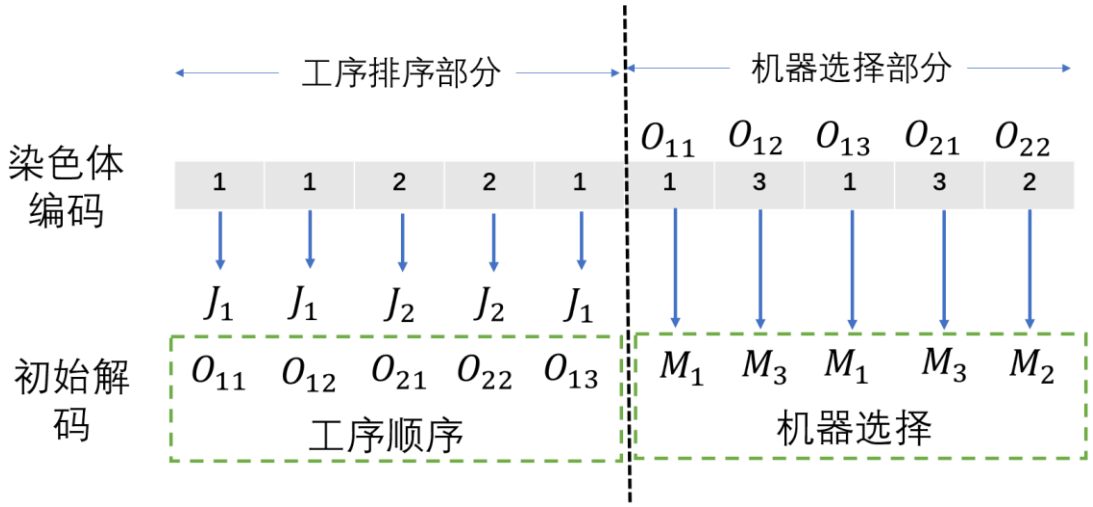


图 3-4-1：染色体编码和解码

4.粒子位置更新策略

基于基本粒子群算法的进化优化机制，可以定义离散粒子群算法的位置更新公式[11]如下：

$$X_i^{k+1} = c_2 \otimes f(c_1 \otimes g(\omega \otimes h(X_i^k), pB_i^k, gB^k) \quad (3\text{-}4\text{-}3)$$

ω 为惯性权重因子，c_1 为认知系数，c_2 为社会系数，$\omega, c_1, c_2 \in [0,1]$ 。

该位置更新公式由以下三个部分构成

(1) $$E_i^k = \omega \otimes h(X_i^k) = \begin{cases} h(X_i^k) & rand < \omega \\ X_i^k & otherwise \end{cases} \quad (3\text{-}4\text{-}4)$$

该部分公式是粒子对先前状态的思考，其中 rand 为[0.1]区间范围内的随机数，$h(X_i^k)$ 表示粒子的速度。

$h(X_i^k)$ 通过粒子的变异操作来实现。$h(X_i^k)$ 使用交换变异方式，经过变异，能够得到一个可行的工序序列，避免了非法解的产生。

(2) $$F_i^k = c_1 \otimes g(E_i^k, pB_i^k) = \begin{cases} g(E_i^k, pB_i^k) & rand < c_1 \\ E_i^k & otherwise \end{cases} \quad (3\text{-}4\text{-}5)$$

该部分公式表示粒子对自生的学习，其中 rand 为[0.1]区间范围内的随机数，$g(E_i^k, pB_i^k)$ 表示的是粒子根据自身最优位置进行的调整。

调整 $g(E_i^k, pB_i^k)$，为了避免非法解的出现。我们基于文献[9]，IPOX 操作的子代可以更好的继承父代，选择 POX 交叉算子进行调整。

(3) $$X_i^k = c_2 \otimes g(F_i^k, gB^k) = \begin{cases} g(F_i^k, gB^k) & rand < c_2 \\ F_i^k & otherwise \end{cases} \quad (3\text{-}4\text{-}6)$$

该部分公式表示染色体粒子根据整体的全局最优的学习。

5.局部搜索策略

我们使用局部搜索策略（LS）来实现提高粒子群算法的精度。在执行完一次粒子位置更新后，对种群中适应度前 10%的个体以及其他任意 20%的个体，进行局部搜索。本文使用基于 Intercharge 邻域结构的局部搜索策略，对相应选中的个体进行局部搜索。设 H 是要执行局部搜索的个体。

（1）产生一个随机整数 d_1 ，$d_1 \in [1, len]$,len 表示粒子的长度。设 $m = d_1$ ，$X'_h = X_h$.下面有一半的几率执行（2），一半的几率执行（3）。

(2) 向前搜索 2.1 若 $d_1 > 1$ ，随机产生范围为 $[0, d_1 - 1]$ 的自然数 d_2 ，表示从 X_h 上第 d_1 位基因开始向前搜索 d_2 位。

2.2 交换 $X_h^{'}$ 上第（m-1）位和 d_2 位的基因值。若 $f(X_h^{'}) < f(X_h)$ ，表示交换后适宜度提高，设置 $X_h = X_h^{'}$

2.3 m=m-1.若 $m > (d_1 - d_2)$ ，返回 2.2

(3) 向后搜索 3.1 若 $d_1 < len$ ，随机产生范围为 $[0, len - d_1]$ 的自然数 d_2 ，表示从 X_h 上第 d_1 位基因开始向后搜索 d_2 位。

2.2 交换 $X_h^{'}$ 上第（m+1）位和 d_2 位的基因值。若 $f(X_h^{'}) < f(X_h)$ ，表示交换后适宜度提高，设置 $X_h = X_h^{'}$

2.3 m=m+1.若 $m < (d_1 - d_2)$ ，返回 2.2

3.4.3 改进的遗传算法

对于两道及以上工序，遗传算法是求解此类组合优化问题的有效方法，考虑到遗传算法易早熟，易于陷入局部最优的弊端，我们使用改进的遗传算法。在遗传算法的结构下，对优化目标使用新的初始化方法，希望能够改善求解速度和到达全局最优解。对于种群，结合小生境的思想，将种群划分成优质子种群和次优子种群。

1.染色体编码和解码：

遗传算法的编码与粒子群相同，见上文。

2.改进初始化方法减少迭代次数

初始解会影响算法的收敛速度和求解的质量。使用的较多的方法是随机初始化.在这种情况下，初始解良莠不齐，需要增加种群规模和迭代次数来得到较优的解。我们首先使用考虑到工序加工时间和运输时间的最短路径算法产生较优的初始解，缩短收敛所需时间。为保证种群多样性，初始化一半采用较优的解，一半使用随机初始化方法生成。

3.交叉算子

通过交叉算子，实现优良父代个体之间的信息交换，产生新的子代。染色体的两部分采用不同的交叉方式。

(1)工序染色体：该部分使用文献[9]提出的 IPOX 交叉法，来继承父代的优良基因，交叉过程如图三所示。P1 和 P2 是父代染色体，C1 和 C2 为子代。过程为：把所有工件集随机分成两个子集 J1 和 J2，复制 P1 中属于 J1 的工件到 C1，复制 P2 包含在 J2 的工件到 C·2，保留位置。复制 P2 包含在 J2 的工件到 C1，P1 包含在 J1 的工件到 C2，保留顺序不变。

C_1	1	2	3	1	1	2	2	3	3
	↑		↑	↑	↑			↑	↑
P_1	1	2	3	1	1	2	2	3	3
P_2	2	1	3	1	2	3	2	1	3
	↓				↓		↓		
C_2	2	1	3	1	2	1	2	3	3

图三：IPOX 交叉过程图解

（2）机器染色体：使用 MPX 方式对基因进行多点交叉，保证此部分交叉后还是可行解，随机选取多个交叉点，两父代进行基因块互换。

4.变异算子

变异是通过随机改变染色体的基因，产生新的染色体，改善遗传算法的局部搜索能力。

（1）工序染色体：随机选择位置进行互换。

（2）机器染色体：随机选择一个基因，使用另外的可选机器代替当前机器。

5.小生境进化策略

在传统遗传算法中，随着进化的进行，优秀个体的重抽样机会上升，容易使得结果过早陷入局部最优的情况。所以本题将通过设定一个小生境，按照适应度值将较优解保留到精英库，将种群划分两个部分，结合遗传算法的交叉变异更新得到一个新一代种群。

从概率上讲,与最优解有较大相似度的个体有较高的适应度[6]，小生境隔离策略以当前最优点为中心，将种群划分为两个部分。小于阈值 d 表明与最优解有较高的相似度，划分到优质子种群 Population1;其余个体划分到次优子种群 Population2。其中子种群 Population1 为提高整体适应度，子种群 Population2 来保持种群的多样性。

我们使用汉明距离来表示与最优解的相似度，汉明距离小的被划入优质子种群，大的被划入次优子种群。汉明距离的计算如下:

$$H(C_1,C_2)=\sum_{i=1}^{n} op1(i)\oplus op2(i)+\sum_{j=1}^{m} mc1(j)\oplus mc2(j)$$

式中，C_1 和C_2 表示两个父代，op1 和 op2 是工序编码部分，mc1 和 mc2 是机器分配编码部分；m 和 n 分别为工序部分和机器分配部分 de 染色体长度。⊕ 为异或运算符。

根据汉明距离将种群平均分为两个部分，分别为优质种群和次优种群。优质种群使用较小的交叉和变异率，以求得局部最优解，次优种群使用较大的交叉和变异率，以探索整个空间。

6.执行步骤

（1）确定种群规模、迭代次数、交叉概率以及变异概率等参数;

（2）种群初始化，一半随机初始化，一半采用较优解。

（3）评价染色体适应值，找到当前最优解，与历史最优解比较，如更好，则取代。将优质染色体放入精英库。

（4）计算种群所有点和最优点的距离，将种群划分成优质种群和次优种群。

（5）使用锦标赛选择策略，分别将两个子种群产生子代。

（6）优质子代交叉变异，交叉概率和变异概率较小；次优子代交叉变异，交叉概率和变异概率较大;

（7）将精英库放回种群；判断是否满足结束条件，若否，则执行（3)；

四、问题二

4.1 问题分析

对于工件只有一道工序的情况下，我们选用最短时间算法来生成初始的一半解，使用随机方法生成另外的一半解，保证离散粒子群的多样性，缩短算法收敛寻优的时间。使用离散粒子群算法来寻找较优的解。

4.2 模型的建立

$$C_{\max}=\min(\max_{1\le j\le n}(C_j))$$

$$s.t.\begin{cases} C_{ijh}\ge P_{ijh}+S_{ijk} \\ C_{ijh}-C_{ij'h'}\ge P_{ijk} \\ S_{ijh}=\begin{cases} C_{ej(h-1)}+T_{ie} & C_{ij'h'}<C_{ej(h-1)}+T_{ie} \\ C_{ij'h'} & C_{ij'h'}>C_{ej(h-1)}+T_{ie} \end{cases} \\ S_{ihj}\ge C_{i'h'j}+T_{ii'}+N_i \\ p=1,2,...,m \\ k=1,2,...,n \\ h=1,2 \end{cases}$$

4.3 模型的求解

4.3.1 求解算法流程

1.设置初始参数。初始化离散粒子群算法常量，惯性权重ω ，认知系数c_1 和社会系数c_2 ，

确定最大迭代次数N_p 和种群规模 G;

1.基于工序表达法的编码方法，使用短时间算法来生成初始的一半解，使用随机方法生成另外的一半解。进行算法的初始化，对种群中所有粒子的位置和速度进行初始化。

2.根据问题的适应度函数，对每个粒子的适应度进行计算评价。

3.将种群中每个粒子的适应度值与它自身所经历过的最佳位置 pBi 的适应度值进行比较，如更好，则替换当前的最优。将所有粒子的最优位置与全局最优进行比较，如更好，则替换当前的全局最优。

4. 根据式 3-4-1 和 3-4-2 更新各个粒子的位置和速度。

5.判断是否满足算法的终止条件，是，则输出当前最优解，否则，转向 2.

算法的流程图如图 4-3-1 所示。算法的具体代码见附录代码 4-3-1。

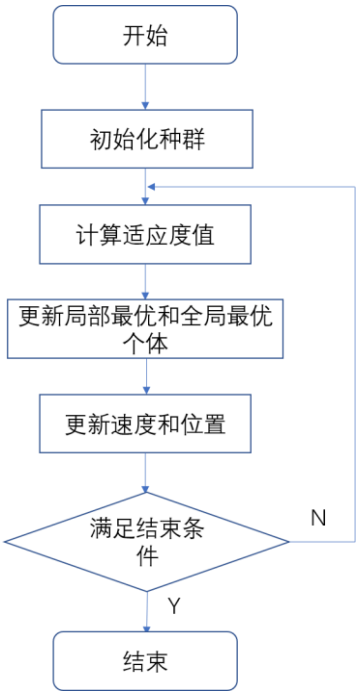


图 4-3-1

4.3.2 求解结果

1.在一道工序，没有故障的情况下。在三组数据下，求解的结果如下，求解的 RGV 的调度策略见附件 2 的 excel 表格。

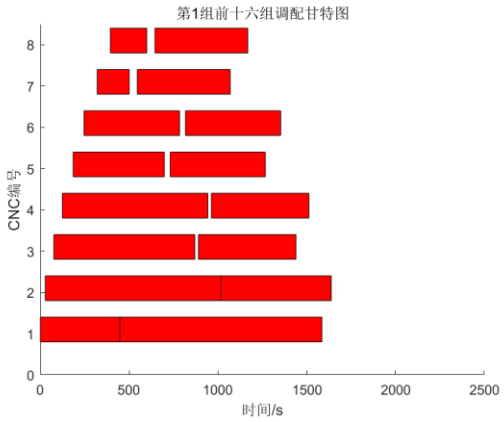
数据	完成的物料	实际下线的物料	系统的作业效率
第一组	377	370	46.25
第二组	326	320	40.00
第三组	386	380	47.50

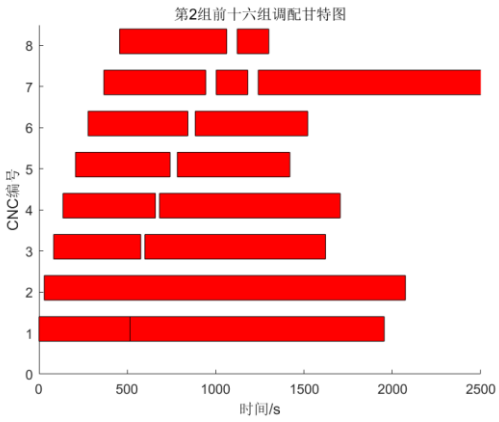
2.在一道工序，存在故障，故障率为 1%，故障排除时间服从[10,20]之间的均匀分布的情况下。在三组数据下，求解的结果如下，求解的 RGV 的调度策略见附件 2 的 excel 表格。

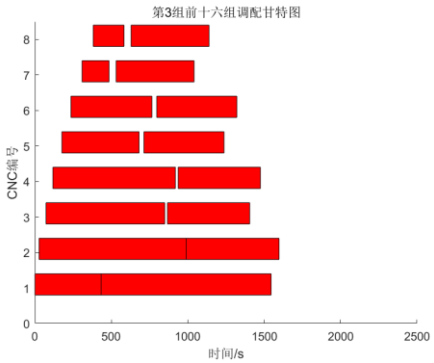
数据	完成的物料	实际下线的物料	系统的作业效率
第一组	375	367	45.875
第二组	317	309	38.625
第三组	362	356	44.500

4.3.3 仿真

1．在一道工序没有故障的情况下，分别对生成的 Excel 表格 Case_1_result 中的三组数据的前十六组数据进行仿真得到其前十六组数据的甘特图。生成甘特图代码见附录代码 4-3-3。

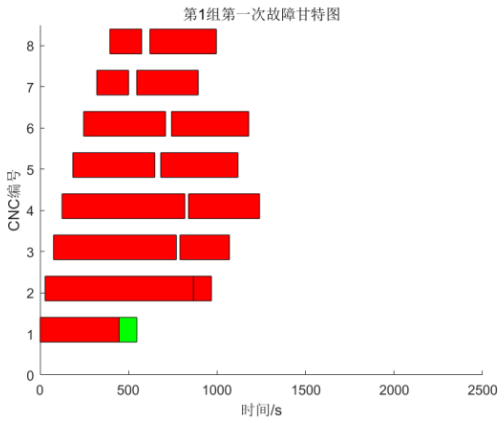


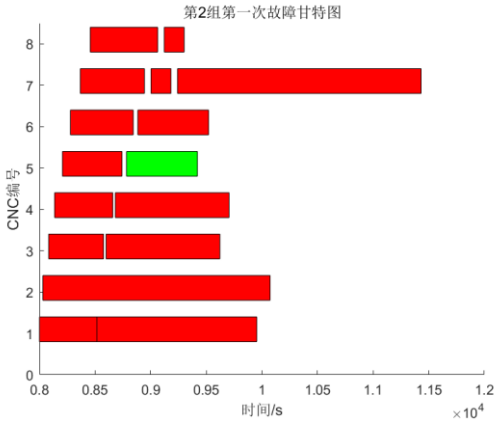


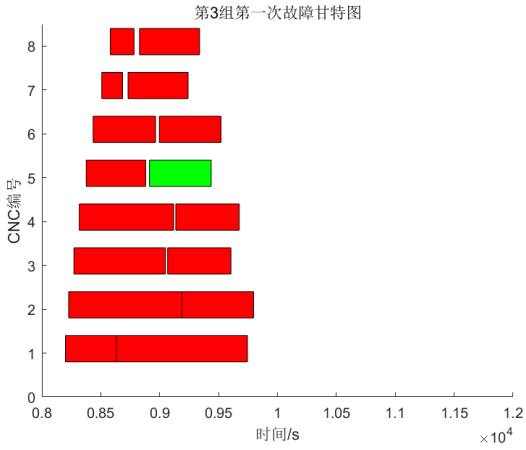


图形说明：每个矩形代表一个任务，从左至右矩形的左边代表任务的开始即开始上料时间，矩形的右边代表任务的结束即开始下料时间。

2. 在一道工序存在故障概率为 1%的情况下，对生成的 Excel 表 Case_3_result_1 的第一次故障部分进行仿真得到对应的甘特图。并用绿色部分表示故障。







五、问题三

5.1 问题分析

对于两道工序的考虑运输时间和上下料时间的柔性车间调度模型，我们使用遗传算法求解此类组合优化问题。考虑到遗传算法易早熟，易于陷入局部最优的弊端，我们使用改进的遗传算法。在遗传算法的结构下，对优化目标使用新的初始化方法，希望能够改善求解速度和到达全局最优解。对于种群，结合小生境的思想，将种群划分成优质子种群和次优子种群。考虑两道工序在 CNC 上的分配，本文取 CNC 编号为 1，3，5，7 的执行第一道工序，为 2，4，6，8 的执行第二道工序。

5.2 模型的建立

$$C_{\max}=\min(\max_{1\le j\le n}(C_j))$$

$$s.t.\begin{cases} C_{ijh}\ge P_{ijh}+S_{ijk} \\ C_{ijh}-C_{ij'h'}\ge P_{ijk} \\ S_{ijh}=\begin{cases} C_{ej(h-1)}+T_{ie} & C_{ij'h'}<C_{ej(h-1)}+T_{ie} \\ C_{ij'h'} & C_{ij'h'}>C_{ej(h-1)}+T_{ie} \end{cases} \\ S_{ihj}\ge C_{i'h'j}+T_{ii'}+N_i \\ p=1,2,...,m \\ k=1,2,...,n \\ h=1,2 \end{cases}$$

5.3 模型的求解

1.在两道工序，没有故障的情况下。在三组数据下，求解的结果如下，求解的 RGV 的调度策略见附件 2 的 excel 表格。求解代码见附录代码 5-3

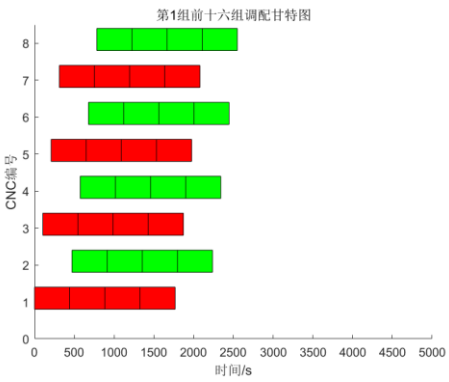
数据	完成的物料	实际下线的物料	系统的作业效率
第一组	253	253	31.625
第二组	209	209	26.125
第三组	232	232	29.000

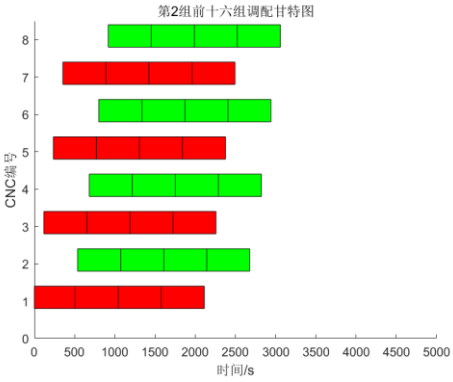
2．在两道工序，存在故障且故障概率为 1%时。在三组数据下，求解的结果如下，求解的 RGV 的调度策略见附件 2 的 excel 表格。

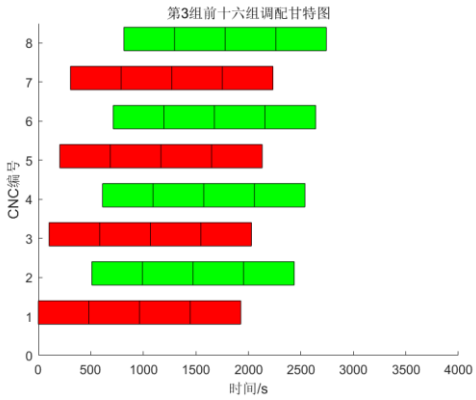
数据	完成的物料	实际下线的物料	系统的作业效率
第一组	249	249	31.125
第二组	204	204	25.5
第三组	228	227	28.375

5.4 仿真

1.在两道工序，没有故障的情况下。在三组数据下，仿真得到对应的甘特图（其中红色为第一道工序，绿色为第二道工序)。







六、模型的优缺点及改进方向

6.1 模型的优缺点

6.1.1 模型的优点

（1）由于题目的条件给定，相比于文献[7]来说多考虑了 RGV 的运输时间，同样的相比于大多数柔性车间调度的文献多出了对于上下料时间和机器故障的考虑。

（2）相比于参考文献中的许多方法如：利用传统遗传算法求解考虑运输时间因素影响的柔性车间调度问题[8]和借助遗传算法和禁忌搜索算法解决了考虑运输时间的柔性调度问题[9]，本模型考虑到了遗传算法和禁忌搜索算法的易于陷入局部最优的缺点。所以在遗传算法的基础上结合小生境进化策略划分种群进行局部寻优同时平衡种群。并且针对优化的目标采用了新的初始化方法来提高求解速度。

6.1.2 模型的缺点

（1）本模型只是单纯地考虑在单个 RGV 工作时的车间调度问题，并没有考虑到多个 RGV 协同工作的情况。事实上在文献《含有 AGV 的柔性车间调度优化研究》[10]中作者在粒子群算法的基础上改进加入了遗传算法证明了当整个加工周期越长，增加 AGV（在本文中为 RGV）的数量效果会更好。其实是随着 RGV 的数量增加 CNC 的等待时间就会大大减少。

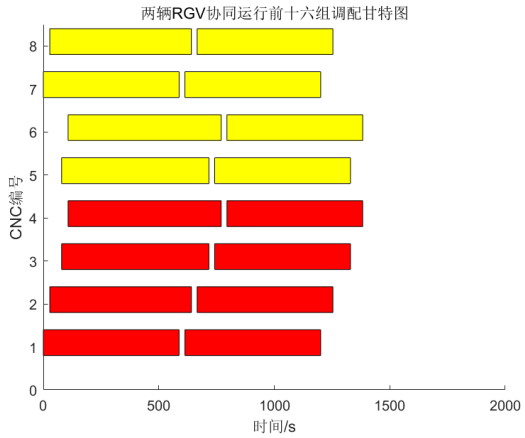
6.2 模型的改进

题目所讨论的是只有一辆 RGV 小车的情况，对于该模型更深入地讨论，可以推广至多辆小车的情况，比如就该问题的只有一个加工工序的情况，可以假设有两辆小车，在 RGV 直线轨道上通行，

要求作业信号发出后各自执行命令，互不影响。

对于两辆小车，各自的工作状况互不影响，并且两辆车工作顺序不可以产生碰撞。对于单工序模型，可以求解出单台 RGV 工作 8 个小时，在问题中所给的情况 1 的条件下，两辆 RGV 小车可以完成的工件总数为 380 件，比单辆 RGV 车负责所有的 CNC 机器情况略微提升了一点。影响十分小的原因是单辆 RGV 单工序情况得到的解已经十分高效，8 小时总时间内，工作时间的占比已经相当大，而新的 RGV 小车的加入，仅仅提高了小车到达目标 CNC 机器的灵活度，路程上花费的时间远小于物料加工的时间。解决该问题所用的算法是结合了粒子群与遗传算法的禁忌搜索算法，该算法是一种全局逐步寻优的算法，在求解问题时，既能够保证多样性，又能够达到全局最优。所以采用禁忌算法求解该问题。由于初始解对于禁忌搜索算法来说特别重要，对其效率有很大影响，为了得到质量较好的初始解，解决该问题的 matlab 代码见附录 6-2-1。

对于在一根 RGV 导轨上的两辆 RGV 车 RGV1 和 RGV2, 由两辆 RGV 小车对 8 台 CNC 进行服务，两辆 RGV 除去交换位置外可在 RGV 导轨上自由移动并服务 CNC。下图是对其前十六个工件的仿真甘特图。



注：红色部分为 RGV1 负责，黄色部分为 RGV2 负责。

类似的，可以研究加工两道工序的情况，也就是两辆 RGV 小车来处理问题中的情况 2，完成问题中情况 2，两道工序的任务，总共可以完成工件数为 276 个，这相比较原来单辆小车完成工件数为 253 个的情况有一些改善。分析其中原因，两道工序的情况相对来说比较复杂，流程多，对于系统的灵活性要求更加高。因此两道工序的情况可以通过两辆 RGV 小车得到一定的改善。解决该问题的 matlab 代码见附录 6-2-2

七、参考文献

[1] 杨立熙，余慧慧. 考虑运输时间的柔性作业车间调度问题研究[J]. 武汉理工大学学报(信息与管理工程版), 2017，第 39 卷(1):608-613.

[2] 马丽丽，基于改进粒子群算法的车间作业调度 哈尔滨理工大学 2010 年

[3]赵宁，李开典，田青 考虑运输时间柔性作业车间调度问题的快速寻优方法 [J]. 计算机集成制造系统，2015,21（3）：724 – 732

[4]金菊良.杨晓华.丁晶.标准遗传算法的改进方案：加速遗传算法 [J]. 系统工程理论与实践，2001,21（4）：8 – 13 .

[5] 何利.刘永贤.谢华龙.刘笑天.基于粒子群算法的车间调度与优化.东北大学.2008

[6]张国辉.柔性作业车间调度方法研究 [D]. 武汉：华中科学技术大学.2009.

[7]李峥峰.多时间因素作业车间调度问题的研究与工程应用 [D]. 武汉：华中科学技术大学, 2019.

[8]ZHANG Q, MANIER H, MANIER M A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times [J]. Computers and Operations Research,2012,39(7) ： 1713 -1723.

[9] 张超勇.基于自然启发式算法的作业车间调度问题理论与应用研究[D].华中科技大学,2006. DOI:10.7666/d.d092918.

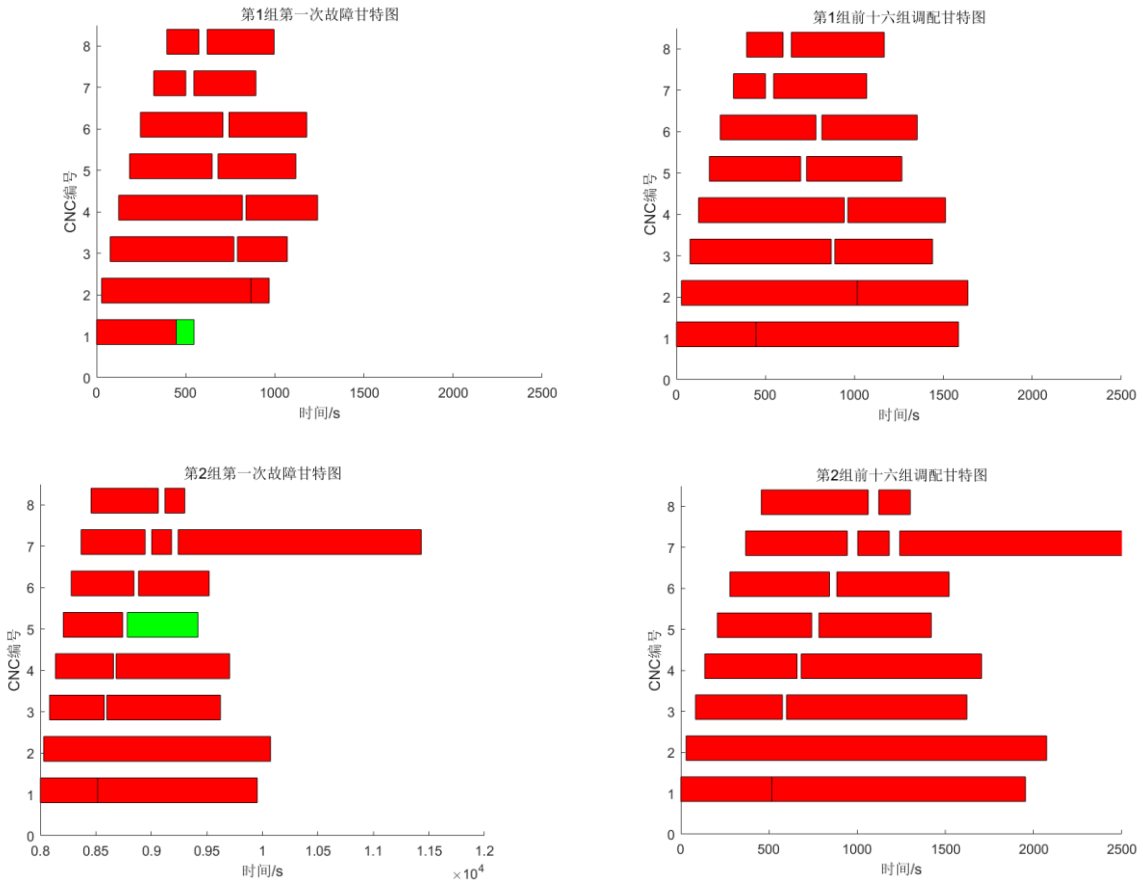
[10]徐云琴，叶春明，曹磊 含有 AGV 的柔性稽查你调度优化研究 上海理工大学，2017

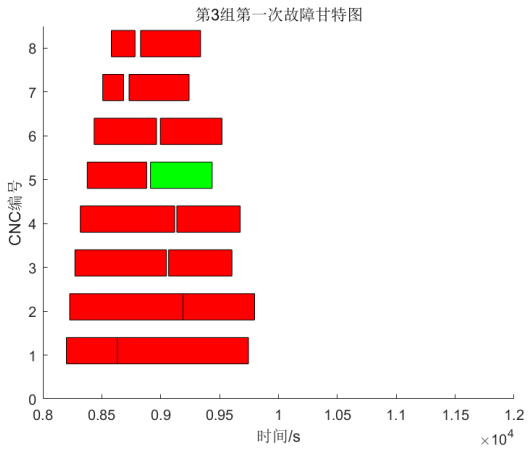
[11]潘全科,王文宏,朱剑英,赵保华.基于粒子群优化和变邻域搜索的混合调度算法[J].计算机集成制造系统,2007(02):323-328.

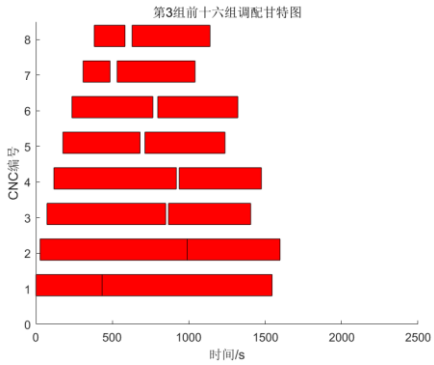
[12] 王磊.基于改进离散粒子群算法的作业车间调度方法的研究及应用[D].浙江工业大学,2012. DOI:10.7666/d.y2142598.

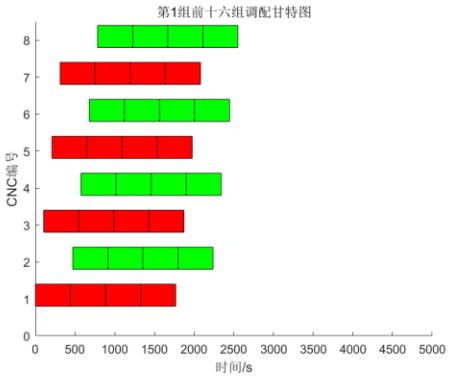
八、附录

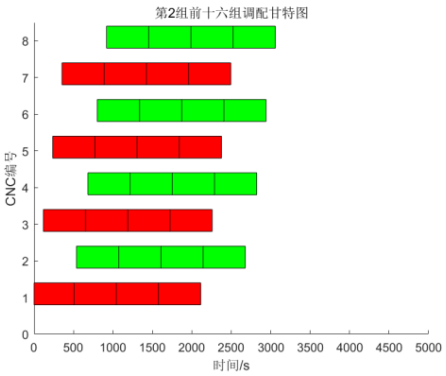
图 4-3

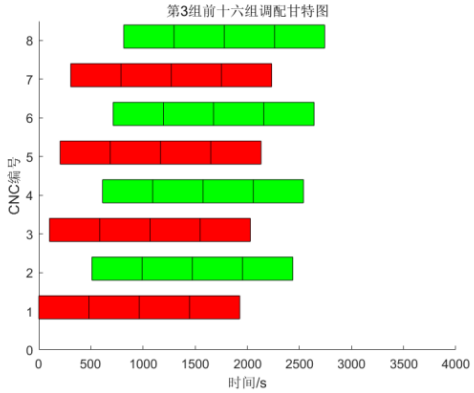












注：甘特图，从左向右每个矩形框的开始是上料开始时间，矩形框的结束时下料开始时间。有故障的甘特图，故障用绿色表示，时间随机赋值。

代码 4-3-1

无故障的：

```
clear;
move1=20;
move2=33;
move3=46;
```

```
machine_time=[0 0 0 0 0 0 0 0];
load1=28;
load2=31;
process_time=560;
wash_time=25;
machine_busy=[0 0 0 0 0 0 0 0];
%移动时间
arrive_time(1,:)=[0 0 move1 move1 move2 move2 move3 move3 ];
arrive_time(2,:)=[0 0 move1 move1 move2 move2 move3 move3 ];
arrive_time(3,:)=[move1 move1 0 0 move1 move1 move2 move2 ];
arrive_time(4,:)=[move1 move1 0 0 move1 move1 move2 move2 ];
arrive_time(5,:)=[move2 move2 move1 move1 0 0 move1 move1 ];
arrive_time(6,:)=[move2 move2 move1 move1 0 0 move1 move1 ];
arrive_time(7,:)=[move3 move3 move2 move2 move1 move1 0 0 ];
arrive_time(8,:)=[move3 move3 move2 move2 move1 move1 0 0 ];
%移动加上下料
arrive_timep(1,:)=[0+load1 0+load2 move1+load1 move1+load2 move2+load1 move2+load2
move3+load1 move3+load2 ];
arrive_timep(2,:)=[0+load1 0+load2 move1+load1 move1+load2 move2+load1 move2+load2
move3+load1 move3+load2 ];
arrive_timep(3,:)=[move1+load1 move1+load2 0+load1 0+load2 move1+load1 move1+load2
move2+load1 move2+load2 ];
arrive_timep(4,:)=[move1+load1 move1+load2 0+load1 0+load2 move1+load1 move1+load2
move2+load1 move2+load2 ];
arrive_timep(5,:)=[move2+load1 move2+load2 move1+load1 move1+load2 0+load1 0+load2
move1+load1 move1+load2 ];
arrive_timep(6,:)=[move2+load1 move2+load2 move1+load1 move1+load2 0+load1 0+load2
move1+load1 move1+load2 ];
arrive_timep(7,:)=[move3+load1 move3+load2 move2+load1 move2+load2 move1+load1
move1+load2 0+load1 0+load2 ];
arrive_timep(8,:)=[move3+load1 move3+load2 move2+load1 move2+load2 move1+load1
move1+load2 0+load1 0+load2 ];
%初始位置
AGV_position=1;
%now_arrive_time;
%物料
number=378;
metal=zeros(3,number);
total_time=0;
machine_metal=zeros(1,8);
for loop=1:number
now_arrive_time=(arrive_time(AGV_position,:));
%cha=now_arrive_time-machine_time;
cha=machine_time-now_arrive_time;
```

```
%判断是否有小于 0 的
minszero=0;
for ii=1:8
    if cha(ii)<=0
        minszero=1;
    end
end
xx=zeros(1,8);
xx(1,:)=inf;
need_wait=0;
if minszero==1%如果有小于零的
    for ii=1:8
        if cha(ii)<=0
            xx(ii)=arrive_timep(AGV_position,ii);
        end
    end
    [~,next_position]=min(xx);
    needwait=0;
else
    [wait_time,next_position]=min(cha);
    needwait=1;
end
%移动
AGV_position=next_position;
disp(next_position);
comsume_time=arrive_timep(next_position);
machine_time=machine_time-comsume_time;%预期完成时间减少
%total_time=total_time+arrive_time(next_position);
if need_wait==1
    machine_time=machine_time-wait_time;
    total_time=total_time+wait_time;
end
machine_time(next_position)=machine_time(next_position)+process_time;%上料的机器时间增加
metal(1,loop)=total_time+arrive_time(next_position);

if machine_busy==1
    metal(2,machine_metal(next_position))=total_time;
end
machine_metal(next_position)=loop;
metal(3,loop)=next_position;
total_time=total_time+comsume_time;

machine_busy(next_position)=1;
if machine_busy==1
```

```
        machine_time=machine_time-wash_time;

        total_time=total_time+wash_time;
    end
    end
```

有故障的：

```
    clear;
    move1=18;
    move2=32;
    move3=46;
    machine_time=[0 0 0 0 0 0 0 0];
    load1=27;
    load2=32;
    process_time=545;
    wash_time=25;
    machine_busy=[0 0 0 0 0 0 0 0];
    error_map=zeros(4,100);
    error_number=1;
    %machine_disable=[1 1 1 1 1 1 1 1];
    %移动时间
    arrive_time(1,:)=[0 0 move1 move1 move2 move2 move3 move3 ];
    arrive_time(2,:)=[0 0 move1 move1 move2 move2 move3 move3 ];
    arrive_time(3,:)=[move1 move1 0 0 move1 move1 move2 move2 ];
    arrive_time(4,:)=[move1 move1 0 0 move1 move1 move2 move2 ];
    arrive_time(5,:)=[move2 move2 move1 move1 0 0 move1 move1 ];
    arrive_time(6,:)=[move2 move2 move1 move1 0 0 move1 move1 ];
    arrive_time(7,:)=[move3 move3 move2 move2 move1 move1 0 0 ];
    arrive_time(8,:)=[move3 move3 move2 move2 move1 move1 0 0 ];
    %移动加上下料
    arrive_timep(1,:)=[0+load1 0+load2 move1+load1 move1+load2 move2+load1 move2+load2
move3+load1 move3+load2 ];
    arrive_timep(2,:)=[0+load1 0+load2 move1+load1 move1+load2 move2+load1 move2+load2
move3+load1 move3+load2 ];
    arrive_timep(3,:)=[move1+load1 move1+load2 0+load1 0+load2 move1+load1 move1+load2
move2+load1 move2+load2 ];
    arrive_timep(4,:)=[move1+load1 move1+load2 0+load1 0+load2 move1+load1 move1+load2
move2+load1 move2+load2 ];
    arrive_timep(5,:)=[move2+load1 move2+load2 move1+load1 move1+load2 0+load1 0+load2
move1+load1 move1+load2 ];
    arrive_timep(6,:)=[move2+load1 move2+load2 move1+load1 move1+load2 0+load1 0+load2
move1+load1 move1+load2 ];
    arrive_timep(7,:)=[move3+load1 move3+load2 move2+load1 move2+load2 move1+load1
move1+load2 0+load1 0+load2 ];
```

```
    arrive_timep(8,:)=[move3+load1 move3+load2 move2+load1 move2+load2 move1+load1
move1+load2 0+load1 0+load2 ];
    %初始位置
    AGV_position=1;
    %now_arrive_time;
    %物料
    metal_number=370;
    metal=zeros(3,metal_number);
    total_time=0;
    machine_metal=zeros(1,8);
    for loop=1:metal_number
    now_arrive_time=(arrive_time(AGV_position,:));
    %cha=now_arrive_time-machine_time;
    cha=machine_time-now_arrive_time;
    %判断是否有小于 0 的
    minszero=0;
    for ii=1:8
        if cha(ii)<=0
            minszero=1;
        end
    end
    xx=zeros(1,8);
    xx(1,:)=inf;
    need_wait=0;
    if minszero==1%如果有小于零的
        for ii=1:8
            if cha(ii)<=0
                xx(ii)=arrive_timep(AGV_position,ii);
            end
        end
        [~,next_position]=min(xx);
        needwait=0;
    else
        [wait_time,next_position]=min(cha);
        needwait=1;
    end
    %移动
    AGV_position=next_position;
    disp(next_position);
    comsume_time=arrive_timep(next_position);
    machine_time=machine_time-comsume_time;%预期完成时间减少
    %total_time=total_time+arrive_time(next_position);
    if need_wait==1
        machine_time=machine_time-wait_time;
```

```
        total_time=total_time+wait_time;
end
machine_time(next_position)=machine_time(next_position)+process_time;%上料的机器时间增加
metal(1,loop)=total_time+arrive_time(next_position);

if machine_busy(next_position)==1
        metal(2,machine_metal(next_position))=total_time;
end
machine_metal(next_position)=loop;
metal(3,loop)=next_position;
total_time=total_time+comsume_time;

machine_busy(next_position)=1;
if machine_busy(next_position)==1
        machine_time=machine_time-wash_time;

        total_time=total_time+wash_time;
end
if rand<0.01
        error_map(3,error_number)=total_time;
        waste_poss_time=10:20;
        waste_time=waste_poss_time(randi(11))*60;
        error_map(4,error_number)=total_time+waste_time;
        %metal(2,loop)=0;
        error_map(1,error_number)=next_position;
        error_map(2,error_number)=loop;
        error_number=error_number+1;
        machine_time(next_position)=machine_time(next_position)-process_time+waste_time;
        machine_busy(next_position)=0;
end
end
```

代码 4-3-3

```
%甘特图
clear;
axis([0,2000,0,8.5]);                    %x 轴 y 轴的范围
set(gca,'xtick',0:500:2000); %x 轴的增长幅度
set(gca,'ytick',0:1:8.5);              %y 轴的增长幅度
xlabel('时间/s'),ylabel('CNC 编号');          %x 轴 y 轴的名称
title('第 1 组前十六组调配甘特图');          %图形的标题
n_bay_nb=8;       %total bays     //机器数目
n_task_nb = 16;%total tasks    //任务数目
```

```
%x 轴 对应于画图位置的起始坐标 x
n_start_time=xlsread('Case_1_result_1.xls','sheet1','j2:j17');      %  每个工序的开始时间
%length 对应于每个图形在 x 轴方向的长度
n_duration_time=xlsread('Case_1_result_1.xls','sheet1','h2:h17'); % 每个工序的持续时间
%y 轴 对应于画图位置的起始坐标 y
n_bay_start=xlsread('Case_1_result_1.xls','sheet1','b2:b17');       % 工序数目，即在哪一行画线
%工序号，可以根据工序号选择使用哪一种颜色
n_job_id=[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0];
rec=[0,0,0,0];
color=['r','g','b','c','m','y'];
for i =1:n_task_nb
   rec(1) = n_start_time(i);        %矩形的横坐标
   rec(2) = n_bay_start(i)-0.2;   %矩形的纵坐标
   rec(3) = n_duration_time(i);   %矩形的 x 轴方向的长度
   rec(4) = 0.6;
   txt=sprintf('p(%d,%d)=%d',n_bay_start(i),n_job_id(i)+1,n_duration_time(i));%将机器号，工序号，加工时间连城字符串
    rectangle('Position',rec,'LineWidth',0.5,'LineStyle','-','FaceColor',color(n_job_id(i)+1));%draw every rectangle
end
```

代码 5-3

```
%%%%%%%%%%%%%%%%%%%%%%%%情况 3 结果生成%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %      function max_time=jiema(bianmaA,bianmab)
%    bianmaA=zeros(1,600);%%工件号 工序
   bianmaB=zeros(1,600);%%机器号
%   bianmaA=A(1,:);
%   bianmaB=B(1,:);
   bianmaA=best_op;
%   bianmaB=best_mc;
%
 for i=1:300
%       OP=find(op);
%       k=OP(1,1);
      [~,pair]=find(best_op==i);

      bianmaB(1,pair(1,1))=best_mc(1,2*i-1);
      bianmaB(1,pair(1,2))=best_mc(1,2*i);

end
 gongxu1_m=zeros(1,300);%
```

```
 gongxu1_s=zeros(1,300);%
 gongxu1_x=zeros(1,300);
 gongxu2_m=zeros(1,300);%
 gongxu2_s=zeros(1,300);
 gongxu2_x=zeros(1,300);
 shangliaokaishi=zeros(1,600);
B=zeros(1,100);
A=zeros(1,600);
MT=zeros(1,8);
% bianmaB(1,1)=1;
% bianmaB(1,2)=2;
% bianmaB(1,3)=2;
gongxu1=400;
gongxu2=378;
qinxi=25;
% [m,n]=find(bianmaB==1);
tempt=0;
for k=1:600
    %rgv 移动的时间
    if k~=1
        tempt=tempt+dt(bianmaB(1,k),bianmaB(1,k-1));
    end
    machine=bianmaB(1,k);
    if(tempt<MT(1,machine))
        tempt=MT(1,machine);
    end


        A(1,k)=bianmaA(1,k);
        B(1,k)=bianmaB(1,k);
        n=sum(A(:)==bianmaA(1,k));
%           MT(1,machine)=MT(1,machine)+shangxialiao(machine);
        shangliaokaishi(1,k)=tempt;
        tempt=tempt+shangxialiao(machine);
        MT(1,machine)=tempt;%上料开始时间
                                %如果是第二道工序 上料开始时间


     if n==1
        MT(1,machine)=MT(1,machine)+gongxu1;%加工时间
     else
        MT(1,machine)=MT(1,machine)+gongxu2;
        tempt=tempt+qinxi;%如果是第二道工序下料 还需要清洗
     end
```

```
end
q=1;
for j=1:600
    [m,n]=find(A==A(1,j));%找到工件序号相同的 n 表示同一个物料 不同机器的位置下标
   if (length(n)==2)

    gongxu1_m(1,q)=B(1,n(1,1));%两道工序 对应机器的顺序
    k=B(1,n(1,1));
    gongxu2_m(1,q)=B(1,n(1,2));
    gongxu1_s(1,q)=shangliaokaishi(1,j);%工序 1 上料开始
    B(1,n(1,1))=0;%令第一道工序的机器为 0 下一次查找第一工序对应机器号的时候直接找到
下一个用该机器的位置
    [c,d]=find(B==gongxu1_m(1,q));%找到下一个物料的第一道工序的对应机器号
      if length(d)>=2
      gongxu1_x(1,q)=shangliaokaishi(1,d(1,1));%第一道工序被下料的开始时间 第二道工序上
料时间应该为此时间+移动时间
     else
      if length(d)==1
      gongxu1_x(1,q)=shangliaokaishi(1,d);%第一道工序被下料的开始时间 第二道工序上料
时间应该为此时间+移动时间
      end
     end

     if length(d)>=2

gongxu2_s(1,q)=gongxu1_x(1,q)+dt(bianmaB(1,d(1,1)),bianmaB(1,d(1,2)))+shangxialiao(gongxu1_m(
1,q));
     else
          if length(d)==1
          gongxu2_s(1,q)=shangliaokaishi(1,d);
          end
     end
   %工序 2 的下料时间和工序 1 下料时间求法一样
    B(1,n(1,2))=0;
%        B(1,n(1,1))=0;
    [e,f]=find(B==gongxu2_m(1,q));%找到下一个用该机器的位置
    if length(f)>=2
          gongxu2_x(1,q)=shangliaokaishi(1,f(1,1));
    else
          if length(f)==1
          gongxu2_x(1,q)=shangliaokaishi(1,f);
          end
```

```
        end
        %最后 对于重复出现的工件序号 赋值为 0 下次不再能找到
        A(1,n(1,1))=0;
        A(1,n(1,2))=0;
       q=q+1;
     end

end

   max_time=max(MT);
%       end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     function max_time=jiema(bianmaA,bianmab)
%    bianmaA=zeros(1,500);%%工件号 工序
     bianmaB=zeros(1,600);%%机器号
%    bianmaA=A(1,:);
%    bianmaB=B(1,:);
%    bianmaA=best_op;
%    bianmaB=best_mc;
%
gongxu1=400;
gongxu2=378;
  for i=1:300
%         OP=find(op);
%         k=OP(1,1);
       [~,pair]=find(bianmaA==i);

       bianmaB(1,pair(1,1))=bianmab(1,2*i-1);
       bianmaB(1,pair(1,2))=bianmab(1,2*i);

  end
  shangliaokaishi=zeros(1,600);
B=zeros(1,600);
A=zeros(1,600);
MT=zeros(1,8);
% bianmaB(1,1)=1;
% bianmaB(1,2)=2;
% bianmaB(1,3)=2;

qinxi=30;
% [m,n]=find(bianmaB==1);
tempt=0;
for k=1:600
       %rgv 移动的时间
```

```
        if k~=1
            tempt=tempt+dt(bianmaB(1,k),bianmaB(1,k-1));
        end
        machine=bianmaB(1,k);
        if(tempt<MT(1,machine))
            tempt=MT(1,machine);
        end


            A(1,k)=bianmaA(1,k);
            B(1,k)=bianmaB(1,k);
            n=sum(A(:)==bianmaA(1,k));
%               MT(1,machine)=MT(1,machine)+shangxialiao(machine);
            shangliaokaishi(1,k)=tempt;
            tempt=tempt+shangxialiao(machine);
            MT(1,machine)=tempt;%上料开始时间
                                        %如果是第二道工序 上料开始时间


         if n==1
                MT(1,machine)=MT(1,machine)+gongxu1;%加工时间
         else
                MT(1,machine)=MT(1,machine)+gongxu2;
                tempt=tempt+qinxi;%如果是第二道工序下料 还需要清洗
         end


end


    max_time=max(MT);
     end

N=120;%种群规模
m=236;%工件个数
M=m*2;%总工序
machine=8;%八台机器
A=zeros(N,M);%工序顺序
B=zeros(N,M);%机器选择
best_op=zeros(1,M);
best_mc=zeros(1,M);
%初始解
seed=zeros(1,M);
```

```
for i=1:m
    seed(2*i-1)=i;
    seed(2*i)=i;
end%1122334455
seeds=zeros(1,M-8);

A(1,:)=seed(1,:);
A(2,1:16)=[1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8];
A(3,1:16)=[1 2 3 4 1 2 3 4 5 6 7 8 5 6 7 8];
A(4,1:16)=[1 2 3 4 1 2 3 4 5 6 7 8 5 6 7 8];
for i=9:m
    A(2,2*i-1)=i;
    A(2,2*i)=i;
    A(3,2*i-1)=i;
    A(3,2*i)=i;
    A(4,2*i-1)=i;
    A(4,2*i)=i;
end
for i=5:N
    A(i,:)=seed(randperm(numel(seed)));
end
machine1=1;
for i=1:M
    B(1,i)=machine1;
    machine1=machine1+1;
    if machine1==9
        machine1=1;
    end
end
machine1=1;
for i=1:M
    B(2,i)=machine1;
    machine1=machine1+1;
    if machine1==9
        machine1=1;
    end
end
for i=1:M/2
    B(3,2*i-1)=5;
    B(3,2*i)=6;
end
for i=1:M/2
    B(4,2*i-1)=7;
    B(4,2*i)=8;
```

```
end


for i=5:N
    for j=1:M/2
        jishu=[1 3 5 7];
        B(i,2*j-1)=jishu(randi(4));
        oushu=[2 4 6 8];
        B(i,2*j)=oushu(randi(4));
    end
end
%jiaocha_bianyi(A,B,3,2,0.1,0.1,0.1,m,M);
%种群初始化结束
outcome=zeros(N,2);
outcome(:,:)=inf;%耗时
distance=zeros(1,N);%海明距离矩阵
best=inf;%最佳时间
for a=1:10000
    for i=1:N
        outcome(i,1)=jiema(A(i,:),B(i,:));%评价染色体适应值
    end
    %找当前种群最优解
    [now_best,now_position]=min(outcome(:,1));
    %与历史最优解比较
    if now_best<best
        best=now_best;
        position=now_position;
        disp(best);
        best_op(1,:)=A(position,:);
        best_mc(1,:)=B(position,:);
    end
    %种群所有点与最优点的海明距离
    op1=A(now_position,:);%精英个体
    mc1=B(now_position,:);
    for j=1:N
        op2=A(j,:);
        mc2=B(j,:);
        distance(j)=haiming(op1,op2,mc1,mc2,M);
    end
%     dmax=max(distance);
%     distance(now_postion,1)=inf;
%     dmin=min(distance);
%     distance(now_postion,1)=0;
%     d=(dmax+dmin)/2;
```

```
    %划分成两个子种群
%       pop1count,pop2count=0;
%       for k=1:N
%           if distance(k,1)<d
%               pop1_op(pop1count)=A(k,:);
%               pop1_mc(pop1count)=B(k,:);
%               pop1_outcome(pop1count)=outcome(k,1);
%           else
%               pop2_op(pop2count)=A(k,:);
%               pop2_mc(pop2count)=B(k,:);
%               pop2_outcome(pop2count)=outcome(k,1);
%           end
%       end
    %划分成两个子种群
    %优质种群
    pop1_op=zeros(N/2,M);
    pop1_mc=zeros(N/2,M);
    %次优种群
    pop2_op=zeros(N/2,M);
    pop2_mc=zeros(N/2,M);
    [b,xx]=sort(distance);
    pop1outcome=zeros(N/2);
    pop2outcome=zeros(N/2);
    for i=1:N/2
        pop1_op(i,:)=A(xx(i),:);
        pop1_mc(i,:)=B(xx(i),:);
        pop1outcome(i)=outcome(xx(i),1);
    end
    for i=(N/2+1):N
        pop2_op(i-N/2,:)=A(xx(i),:);
        pop2_mc(i-N/2,:)=B(xx(i),:);
        pop2outcome(i-N/2)=outcome(xx(i),1);
    end
     %pop1

     %锦标赛选择产生 pop1 子代
     pop1_son_op=zeros(N/4,M);
     pop1_son_mc=zeros(N/4,M);
     for i=1:N/4
         x1=randi(N/4);
         x2=randi(N/4);
         if pop1outcome(x1)<pop1outcome(x2)
             if rand<0.8
                 pop1_son_op(i,:)=pop1_op(x1,:);
```

```
            pop1_son_mc(i,:)=pop1_mc(x1,:);
        else
            pop1_son_op(i,:)=pop1_op(x2,:);
            pop1_son_mc(i,:)=pop1_mc(x2,:);
        end
    else
        if rand<0.8
            pop1_son_op(i,:)=pop1_op(x2,:);
            pop1_son_mc(i,:)=pop1_mc(x2,:);
        else
            pop1_son_op(i,:)=pop1_op(x1,:);
            pop1_son_mc(i,:)=pop1_mc(x1,:);
        end
    end
end
%pop2


%锦标赛选择产生 pop2 子代
pop2_son_op=zeros(N/4,M);
pop2_son_mc=zeros(N/4,M);
for i=1:N/4
    x1=randi(N/4);
    x2=randi(N/4);
    if pop2outcome(x1)<pop2outcome(x2)
        if rand<0.8
            pop2_son_op(i,:)=pop2_op(x1,:);
            pop2_son_mc(i,:)=pop2_mc(x1,:);
        else
            pop2_son_op(i,:)=pop2_op(x2,:);
            pop2_son_mc(i,:)=pop2_mc(x2,:);
        end
    else
        if rand<0.8
            pop2_son_op(i,:)=pop2_op(x2,:);
            pop2_son_mc(i,:)=pop2_mc(x2,:);
        else
            pop2_son_op(i,:)=pop2_op(x1,:);
            pop2_son_mc(i,:)=pop2_mc(x1,:);
        end
    end
end

%交叉，变异
```

```
%       AA=zeros(N,M);
%       BB=zeros(N,M);
        for i=1:N/4

[A(i,:),A(i+N/4,:),B(i,:),B(i+N/4,:)]=jiaocha_bianyi(pop1_son_op,pop1_son_mc,randi(N/4),randi(N/4),0.1,
0.1,0.1,m,M);
        end

        for i=N/2+1:N/4*3

[A(i,:),A(i+N/4,:),B(i,:),B(i+N/4,:)]=jiaocha_bianyi(pop2_son_op,pop2_son_mc,randi(N/4),randi(N/4),0.5,
0.5,0.5,m,M);
        end
        %精英放回种群
        A(N,:)=op1;
        B(N,:)=mc1;
    end
    disp('end');
    %小生境隔离策略

    function [ c1,c2,c3,c4 ] = jiaocha_bianyi( A,B,f1,f2,exchange1,exchange2,exchange3 ,m,M)
    %UNTITLED 此处显示有关此函数的摘要
    %   此处显示详细说明
        %交叉算子 pox
        %工序染色体
        p1=A(f1,:);%父代
        p2=A(f2,:);
        seed1=1:m;
        J=seed1(randperm(numel(seed1)));
        job1=J(1:m/2);
        job2=J(m/2+1:m);%工序集随机分成两半
        c1=zeros(1,M);%子代
        c2=zeros(1,M);
        %复制 p1 包含在 job1 的工件到 c1,保留位置
        for i=1:M
            for j=1:m/2
                if p1(i)==job1(j)
                    c1(i)=p1(i);
                end
            end
        end
        %复制 p2 包含在 job2 的工件到 c2，保留位置
        for i=1:M
            for j=1:m/2
```

```
            if p2(i)==job2(j)
                c2(i)=p2(i);
            end
        end
    end

    %复制 p2 包含在 job2 的工件到 c1，保留位置
    for i=1:M
        for j=1:m/2
            if p2(i)==job2(j)
                for k=1:M
                    if c1(k)==0
                        c1(k)=p2(i);
                        break;
                    end
                end
            end
        end
    end

    %复制 p1 包含在 job1 的工件到 c2，保留顺序
    for i=1:M
        for j=1:m/2
            if p1(i)==job1(j)
                for k=1:M
                    if c2(k)==0
                        c2(k)=p1(i);
                        break;
                    end
                end
            end
        end
    end

    %机器染色体 多点交叉
    p3=B(f1,:);
    p4=B(f2,:);
    c3=p3;
    c4=p4;
    exchange=floor(M*exchange1);%交换 10%的基因
    for i=1:exchange
        point=randi(M);
        middle=c3(point);
        c3(point)=c4(point);
```

```
        c4(point)=middle;
    end

    %变异
    %工序排序
    %互换
    for i=1:floor(M*exchange2)%变异 10%
        point1=randi(M);
        point2=randi(M);
        middle=c1(point1);
        c1(point1)=c1(point2);
        c1(point2)=middle;
        point1=randi(M);
        point2=randi(M);
        middle=c2(point1);
        c2(point1)=c2(point2);
        c2(point2)=middle;
    end


    %机器选择
    for i=1:floor(M*exchange3)
        cross1=randi(M);
        cross2=randi(M);
        jishu=[1 3 5 7];
        oushu=[2 4 6 8];
        if c3(cross1)==1||c3(cross1)==3||c3(cross1)==5||c3(cross1)==7
            c3(cross1)=jishu(randi(4));
        else
            c3(cross1)=oushu(randi(4));
        end
        if c4(cross2)==1||c4(cross2)==3||c4(cross2)==5||c4(cross2)==7
            c4(cross2)=jishu(randi(4));
        else
            c4(cross1)=oushu(randi(4));
        end

    end


end
function [ distance ] = haiming( op1,op2,mc1,mc2,M )
%UNTITLED2 此处显示有关此函数的摘要
%    此处显示详细说明
```

```
%小生境隔离策略
    distance=0;
    for i=1:M
        if op1(i)~=op2(i)
            distance=distance+1;
        end
    end
    for i=1:M
        if mc1(i)~=mc2(i)
            distance=distance+1;
        end
    end
end
```

代码 6-2-1

```
% tempt1=0;
% temppos1=1;
% temppos2=4;
% tempt2=0;
tempt=0;
temppos=1;
MT=zeros(1,4);
jiagong=560;
qinxi=25;
yidong=20;
N=190;
mc=zeros(1,N);
op=zeros(1,N);
sl=zeros(1,N);
xl=zeros(1,N);
for i=1:N
    %%%%%
    if temppos==1   %在第一个位置 看 1 2 两台机是否空闲
        if(tempt>=MT(1,1))
            mc(1,i)=1;
            sl(1,i)=tempt;
            tempt=tempt+shangxialiao(1);
            if MT(1,1)~=0
                tempt=tempt+qinxi;
            end
```

```
    MT(1,1)=MT(1,1)+shangxialiao(1)+jiagong;
    continue;
end
if(tempt>=MT(1,2))
    tempt=tempt+shangxialiao(2);
    if MT(1,2)~=0
        tempt=tempt+qinxi;
    end
    MT(1,2)=MT(1,2)+shangxialiao(2)+jiagong;
     continue;
end
if(tempt+yidong>=MT(1,3))
     tempt=tempt+yidong+shangxialiao(3);
     if MT(1,3)~=0
         tempt=tempt+qinxi;
    end
     MT(1,3)=MT(1,3)+shangxialiao(3)+jiagong;
     temppos=2;
     continue;
end
if(tempt+yidong>=MT(1,4))
     tempt=tempt+yidong+shangxialiao(4);
     if MT(1,4)~=0
         tempt=tempt+qinxi;
    end
     MT(1,4)=MT(1,4)+shangxialiao(4)+jiagong;
      temppos=2;
       continue;
end
[a,b]=min(MT);
if tempt<a
    if b==1||b==2 % 留在原地 等机器空闲
        tempt=MT(1,b)+shangxialiao(b)+qinxi;
        MT(1,b)=MT(1,b)+shangxialiao(b)+qinxi+jiagong;
         continue;

    else
        if tempt+yidong<a     %如果移动过来后 rgv 的时间同样小于最小值 则值得移
动
            tempt=MT(1,b)+shangxialiao(b)+qinxi;
            MT(1,b)=MT(1,b)+shangxialiao(b)+qinxi+jiagong;
            temppos=2;
             continue;
```

```
                end

            end


        end
    end
    %%%%
        %%%%%
    if temppos==2   %在第 2 个位置  看 3 4 两台机是否空闲
        if(tempt>=MT(1,3))
            tempt=tempt+shangxialiao(3);
            if MT(1,3)~=0
                tempt=tempt+qinxi;
            end
            MT(1,3)=MT(1,3)+shangxialiao(3)+jiagong;
            continue;
        end
        if(tempt>=MT(1,4))
            tempt=tempt+shangxialiao(4);
            if MT(1,4)~=0
                tempt=tempt+qinxi;
            end
            MT(1,4)=MT(1,4)+shangxialiao(4)+jiagong;
             continue;
        end
        if(tempt+yidong>=MT(1,1))
             tempt=tempt+yidong+shangxialiao(1);
             if MT(1,1)~=0
                 tempt=tempt+qinxi;
            end
             MT(1,1)=MT(1,1)+shangxialiao(1)+jiagong;
             temppos=1;
             continue;
        end
        if(tempt+yidong>=MT(1,2))
             tempt=tempt+yidong+shangxialiao(2);
             if MT(1,2)~=0
                 tempt=tempt+qinxi;
            end
             MT(1,2)=MT(1,2)+shangxialiao(2)+jiagong;
              temppos=2;
               continue;
        end
```

```
            [a,b]=min(MT);
            if tempt<a
                if b==3||b==4 % 留在原地 等机器空闲
                    tempt=MT(1,b)+shangxialiao(b)+qinxi;
                    MT(1,b)=MT(1,b)+shangxialiao(b)+qinxi+jiagong;
                     continue;

                else
                     if tempt+yidong<a     %如果移动过来后 rgv 的时间同样小于最小值 则值得移
动
                        tempt=MT(1,b)+shangxialiao(b)+qinxi;
                        MT(1,b)=MT(1,b)+shangxialiao(b)+qinxi+jiagong;
                        temppos=1;
                         continue;

                     end

                end


            end
        end
        %%%%




    end
```

代码 6-2-2

```
    tempt=0;
    temppos=1;
    MT=zeros(1,4);
    jiagong1=400;
    jiagong2=378;
    qinxi=25;
    yidong=20;
    N=100;
    aa=0;%1gongxu 1weizhi
    ab=0;%1gongxu 2weizhi
    ba=0;%2gongxu 1weizhi
    bb=0;%2gongxu 2weizhi
```

```
for i=1:2*N
 if temppos==1
          if(mod(i,2)==0)
           %%%如果是第二道工序了
            if tempt>=MT(1,2)
                 MT(1,2)=tempt+shangxialiao(2)+qinxi+jiagong2;
                 tempt=tempt+qinxi+shangxialiao(2);
%                  MT(1,2)=te+shangxialiao(2)+jiagong2;
                ab=ab+1;
                continue;
            end

           %%%
            if tempt<MT(1,2)
                if(tempt+yidong<MT(1,4))
                  if(MT(1,4)<MT(1,2))%值得移动
                       tempt=TM(1,4)+qinxi+shangxialiao(4);
                       MT(1,4)=MT(1,4)+shangxialiao(4)+jiagong2;
                       temppos=2;

                       continue;
                  else%不值得移动了
%                        tempt=tempt+qinxi+shangxialiao(2);
                       tempt=TM(1,2)+shangxialiao(2)+qinxi;
                       MT(1,2)=MT(1,2)+shangxialiao(2)+jiagong2;
                       continue;
                  end
                end

            else        %如果现在 2 号机空闲 就原地上料
                        MT(1,2)=tempt+jiagong2+shangxialiao(2);
                        tempt=tempt+qinxi+shangxialiao(2);
%                        MT(1,2)=MT(1,2)+shangxialiao(2)+jiagong2;
                       continue;
            end   %%%%%%%%%%第二工序结束
          end
            %第一工序的
         if(mod(i,2)~=0)
           %%%如果是第一道工序了
            if tempt>=MT(1,1)
                 MT(1,1)=tempt+shangxialiao(1)+jiagong1;
                 tempt=tempt+shangxialiao(1);
%                  MT(1,1)=MT(1,1)+shangxialiao(1)+jiagong1;
                continue;
```

```
            end

           %%%
            if tempt<MT(1,1)
                if(tempt+yidong<MT(1,3))
                   if(MT(1,3)<MT(1,1))%值得移动
                         tempt=MT(1,1)+shangxialiao(3);
                         MT(1,3)=MT(1,3)+shangxialiao(3)+jiagong1;
                          temppos=2;
                         continue;
                   else%不值得移动了
                         tempt=MT(1,1)+shangxialiao(1);
                         MT(1,1)=MT(1,1)+shangxialiao(1)+jiagong1;
                         continue;
                   end
                end

            else        %如果现在 1 号机空闲 就原地上料
                         tempt=tempt+qinxi+shangxialiao(1);
                         MT(1,1)=tempt+shangxialiao(1)+jiagong1;
                         continue;
            end   %%%%%%%%%%第一工序结束
          end

  end
      if temppos==2

          if(mod(i,2)==0)
           %%%如果是第二道工序了
            if tempt>=MT(1,4)
                 MT(1,4)=tempt+shangxialiao(4)+jiagong2;
                 tempt=tempt+qinxi+shangxialiao(4);
%                  MT(1,4)=MT(1,4)+shangxialiao(4)+jiagong2;
                continue;
            end

           %%%
            if tempt<MT(1,4)
                if(tempt+yidong<MT(1,2))
                   if(MT(1,2)<MT(1,4))%值得移动
                         tempt=MT(1,2)+qinxi+shangxialiao(2);
                         MT(1,2)=MT(1,2)+shangxialiao(2)+jiagong2;
                          temppos=1;
                         continue;
```

```
                else%不值得移动了
                    tempt=MT(1,4)+qinxi+shangxialiao(4);
                    MT(1,4)=MT(1,4)+shangxialiao(4)+jiagong2;
                    continue;
                end
            end

        else        %如果现在 2 号机空闲 就原地上料
                    tempt=MT(1,4)+qinxi+shangxialiao(4);
                    MT(1,4)=MT(1,4)+shangxialiao(4)+jiagong2;
                    continue;
      end   %%%%%%%%%%第二工序结束
    end
      %第一工序的
    if(mod(i,2)==1)
      %%%如果是第一道工序了
       if tempt>=MT(1,3)
           MT(1,3)=tempt+shangxialiao(3)+jiagong1;
          tempt=tempt+qinxi+shangxialiao(3);
%                MT(1,3)=MT(1,3)+shangxialiao(3)+jiagong1;
          continue;
       end

      %%%
       if tempt<MT(1,3)
          if(tempt+yidong<MT(1,1))
            if(MT(1,1)<MT(1,3))%值得移动
                tempt=MT(1,1)+qinxi+shangxialiao(1);
                MT(1,1)=MT(1,1)+shangxialiao(1)+jiagong1;
                 temppos=1;
                continue;
            else%不值得移动了
                tempt=MT(1,3)+shangxialiao(3);
                MT(1,3)=MT(1,3)+shangxialiao(3)+jiagong1;
                continue;
            end
          end

       else        %如果现在 1 号机空闲 就原地上料
                   MT(1,3)=tempt+shangxialiao(3)+jiagong1;
                   tempt=tempt+qinxi+shangxialiao(3);
%                  MT(1,3)=MT(1,3)+shangxialiao(3)+jiagong1;
                continue;
       end   %%%%%%%%%%第一工序结束
```

```
            end



    end

end



function [a]=dt(x,y)
c=floor((x+1)/2);
d=floor((y+1)/2);
a=abs(c-d);
switch(a)
    case 0
        a=0;
    case 1
        a=10;
    case 2
        a=16;
    case 3
        a=23;

end

End
```