

基于总加权完成时间的可重入混合流水车间调度问题

轩 华[†], 李 冰, 罗书敏, 王薛苑

(郑州大学 管理工程学院, 郑州 450001)

摘 要: 研究以最小化总加权完成时间为目标的可重入混合流水车间调度问题(RHFS-TWC), 并构建问题的整数规划模型. 根据模型的特点, 设计基于二维矩阵组的调度解编码方案, 结合NEH启发式算法确定工件初始加工顺序, 生成高质量初始调度解群. 为避免算法陷入早熟及扩大解的搜索空间, 给出IGA的遗传参数自适应调整策略, 最终形成NEH-IGA融合求解策略. 针对不同规模问题分别用传统GA、基于遗传参数自适应调整的IGA、NEH启发式、NEH-IGA算法进行仿真测试, 仿真结果表明NEH启发式和遗传参数自适应动态调整策略的引入有效改善了原有GA的求解能力, NEH-IGA算法在求解RHFS-TWC问题方面优势明显.

关键词: 总加权完成时间; 可重入混合流水车间调度; 运输时间; NEH-IGA算法

中图分类号: TB49

文献标志码: A

Reentrant hybrid flowshop scheduling problem based on total weighted completion time

XUAN Hua[†], LI Bing, LUO Shu-min, WANG Xue-yuan

(School of Management Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: This paper discusses a reentrant hybrid flowshop scheduling problem with the objective of minimizing total weighted completion time. An integer programming model is then developed. The scheduling coding scheme of two-dimensional matrix is designed based on the features of the model. Then, the initial job processing sequence is obtained using NEH heuristic in order to generate the initial scheduling population with high quality. The self-adaptive adjustment strategy of the genetic parameters in improved genetic algorithm(IGA) is proposed to avoid the early maturing of the algorithm and expands the search space. Thus the NEH-IGA algorithm is provided combined with the above improvement. Simulation experiments are performed on different sized problems respectively using the traditional GA, IGA based on the self-adaptive adjustment of genetic parameters, NEH heuristic and NEH-IGA. Numerical results show that the introduction of NEH heuristic and the self-adaptive adjustment strategy of genetic parameters can improve the solution ability of the original GA effectively. The NEH-IGA has more advantage in terms of solving the RHFS-TWC problem.

Keywords: total weighted completion time; reentrant hybrid flowshop scheduling; transportation time; NEH-IGA algorithm

0 引 言

在一些实际生产环境中, 工件以相同的工艺路径经过多道工序进行加工, 至少有一道工序包含多台同构并行机, 且需多次重入此加工系统, 这类结构称为可重入混合流水车间^[1]. 可重入混合流水车间调度(Reentrant hybrid flowshop scheduling, RHFS)问题普遍存在于电子工业中, 如半导体晶圆、印刷电路板和薄膜晶体管-液晶显示面板(TFT-LCD)生产中. RHFS问题比传统HFS更为复杂, 已被证明是NP-hard^[2].

由于存在大量客户在订货时要求交货越快越好, 理想状态是即时交货, 但即时交货很难达到, 如何安排不同客户的交货顺序, 引起了制造业的广泛关注. 针对客户情况差异, 对不同客户的工件完成时间赋予不同的权重, 该权重可表示为一种成本或损失, 求解总加权完成时间最小化是解决上述问题的有效方法. 对于不同机器环境下的总加权完成时间问题, 已有不少国内外学者展开了研究. 在单机环境下, 文献[3]研究了在线并行批调度问题, 针对一般加工时

收稿日期: 2017-07-08; 修回日期: 2017-11-02.

基金项目: 教育部人文社会科学研究基金项目(15YJC630148); 国家自然科学基金项目(U1604150); 郑州大学优秀青年教师发展基金项目(1421326092).

作者简介: 轩华(1979—), 女, 教授, 博士, 从事物流优化与控制等研究; 李冰(1976—), 男, 教授, 博士, 从事物流优化与控制等研究.

[†]通讯作者. E-mail: hxuan@zzu.edu.cn

间情况提出了在线算法,当所有工件有相同加工时间时提出了一个最优算法;文献[4]考虑交货期和优先级约束,提出了一种 $\sqrt{e}/(\sqrt{e}-1)$ 近似算法求解带释放日期和优先级约束的非抢占式单机调度问题.在不相关并行机环境下,文献[5]考虑顺序相关调整时间和交货期约束,提出了3种启发式算法.在流水车间环境下,文献[6]提出了启发式算法.对于HFS问题,文献[7]假定运输能力有限,提出了基于阶段分解的拉格朗日松弛算法;文献[8]则针对零等待问题,提出了结合代理次梯度法的拉格朗日松弛算法.就开放车间而言,文献[9]针对从发光二极管测试的芯片分类操作提炼出的带并行机的4阶段开放车间调度,提出了两种模拟退火算法.

RHFS要求每个加工任务多次重入生产系统进行加工.以最小化Makespan为目标,文献[10]采用分枝定界算法得到最优半置换调度表,提出改进Johnson算法、改进CDS算法以及改进NEH算法,得到了非置换调度的近似解.为最小化Makespan和总拖期,文献[11]提出一种改进的基于教学学习的优化算法;文献[12]提出一种迭代的帕累托贪婪算法;文献[13]提出结合不同局域搜索方法的改进算法.为最小化总加权完成时间,文献[14]提出了基于工件分解的拉格朗日松弛算法.

在激烈的市场竞争中,能够在最短的时间内完成客户的订单可以有效提高一个企业的竞争力.研究总加权完成时间的调度问题将直接影响制造业物流系统的实施,可以缩短产品交付时间.就目前查阅的文献而言,极少有研究探讨基于总加权完成时间的RHFS问题,而且求解这类问题的近似算法也很少.为此,本文研究以总加权完成时间最小化为目标的可重入混合流水车间调度(Reentrant hybrid flowshop scheduling with total weighted completion time, RHFS-TWC)问题.鉴于RHFS-TWC问题的模型为大规模数学规划模型,属于NP-hard问题,利用常规算法直接求解非常困难.因此,本文提出一种嵌入NEH启发式的改进遗传算法(Improved genetic algorithm with NEH heuristic, NEH-IGA),该算法不但在求解质量及速度方面表现突出,同时其求解过程及所得解均能够实现对问题的清晰表述.

1 RHFS-TWC问题模型

RHFS-TWC问题是一类典型的组合优化问题,可以描述为 J 个工件经过 G 道工序加工,且需要重入此加工系统 $(L-1)$ 次,至少有一道工序有两台或两台以上的同构并行机,工件到达每道工序之后选择该工

序空闲的一台机器进行加工,如图1所示.

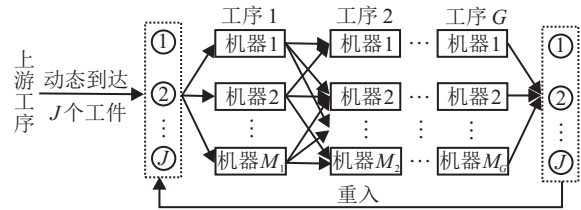


图1 RHFS-TWC调度结构示意图

在某些参数确定的情况下,确定工件的加工顺序和每道工序机器的分配,以实现总加权完成时间最小.由于RHFS可重入系统的特征,处于不同层次的工件可能会互相争夺同一台机器的加工能力,使系统的不稳定性和不确定性增大,其求解难度也较大.

1.1 符号定义

建立模型用到的符号及其含义如下所示:

$\bar{J}$ 为工件集合, $\bar{J} = \{j | j = 1, 2, \dots, J\}$, j 为工件编号, J 为工件总数.

$\bar{G}$ 为工序集合, $\bar{G} = \{g | g = 1, 2, \dots, G\}$, g 为工序编号, G 为工序总数.

$\bar{L}$ 为层次集合, $\bar{L} = \{l | l = 1, 2, \dots, L\}$, l 为层次编号, L 为层次总数,层次集合即对应初始层和各重入层,第1层为初始层,第 l 层则对应重入次数为 $l-1$ 的重入层.

$\bar{S}$ 为时间集合, $\bar{S} = \{s | s = 1, 2, \dots, S\}$, s 为某时刻, S 为计划时间范围.

$\bar{M}_{gl}$ 为机器集合, $\bar{M}_{gl} = \{m_{gl} | m_{gl} = 1, 2, \dots, M_g\}$, m_{gl} 为机器编号, M_g 为 g 工序可使用的机器总数.

P_{igl} 为工件 j 在第 l 层第 g 道工序的加工时间.

W_j 为工件 j 的权重.

R_j 为工件 j 到达加工系统初端的时间.

$Y_{l,g,g+1}$ 为工件在相邻两阶段间的运输时间,当 $g=0$ 时,若 $l>1$,则 $Y_{l,0,1}$ 表示工件由 $l-1$ 层第 G 道工序运输到 l 层的第1道工序的运输时间,其值大于0;否则,令 $Y_{l,0,1}=0$.

X_{jgl} 为0-1变量,若时刻 s 工件 j 正在 l 层第 g 道工序进行加工,则 $X_{jgl}=1$;否则 $X_{jgl}=0$.

C_{jgl} 为工件 j 在第 l 层第 g 道工序的完成时间.

1.2 模型构建

1) 目标函数为

$$\min O = \sum_{j=1}^J w_j C_{jGL}. \quad (1)$$

2) 机器能力约束为

$$\sum_{j=1}^J \sum_{l=1}^L X_{jgl} \leq M_g, \forall g \in \overline{G}, s \in \overline{S}. \quad (2)$$

3) 工序优先级约束为

$$P_{jgl} + Y_{l,g-1,g} \leq C_{jgl} - C_{j,g-1,l}, \\ \forall j \in \overline{J}, g \in \overline{G}, l \in \overline{L}. \quad (3)$$

其中

$$C_{j,0,l} = \begin{cases} C_{j,G,l-1}, & l > 1; \\ 0, & \text{否则}. \end{cases}$$

约束(3)表示只有工件在该层的前一工序或前一层最后一道工序加工完成并运输到下一工序之后,才能开始该工序的加工。

4) 工序加工时间约束为

$$\sum_{s=1}^S X_{jgl} = P_{jgl}, \forall j \in \overline{J}, g \in \overline{G}, l \in \overline{L}. \quad (4)$$

约束(4)表示工件占用机器的时间区间。

5) 工件动态到达约束为

$$C_{j,1,1} + 1 \geq R_j + P_{j,1,1}, \forall j \in \overline{J}. \quad (5)$$

约束(5)表示工件开始加工的时间不能早于其到达第1层第1道工序的时间。

6) 变量 C_{jgl} 的取值范围为

$$C_{jgl} \geq 0, \forall j \in \overline{J}, g \in \overline{G}, l \in \overline{L}. \quad (6)$$

7) 变量 X_{jgl} 的取值范围为

$$X_{jgl} \in \{0, 1\}, \forall j \in \overline{J}, g \in \overline{G}, l \in \overline{L}, s \in \overline{S}. \quad (7)$$

2 RHFS-TWC系统的二维矩阵编码方案

RHFS-TWC调度模型不仅要确定工件加工顺序,而且需要为每道工序分配一台合适的机器。鉴于RHFS-TWC问题的初始层-重入层加工特性,本文设计基于平行机机器号的二维矩阵编码方案进行机器的分配,二维矩阵编码方案的设计是NEH-IGA融合求解算法设计的关键所在。

2.1 基于平行机组的工件加工机器流生成机制

1) 基于平行机机器号的加工机器流。

工件依次在第 l 层各道工序的同构平行机组内随机选择一台空闲机器进行加工,从而形成工件加工机器流,如图2所示,其中 $l \in \overline{L}$ 。

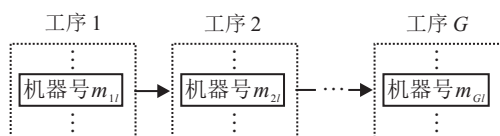


图2 工件加工机器流

2) 基于机器空闲原则的加工流生成机制。

基于同构平行机组内空闲机器才能被选择的原

则,依次筛选各道工序内的机器。如果没有空闲平行机,则工件依次进行等待,有空闲机器时再进行选择。

2.2 工件初始层-重入层加工系统的矩阵编码

使用基于平行机机器号的工件加工机器流表述方法对工件的初始层-重入层加工系统进行编码表述。

1) 初始层编码。

采取基于平行机机器号的工件加工机器流生成机制对初始层进行编码,作为矩阵的第1列列向量。

2) 重入层编码。

采取基于平行机机器号的工件加工机器流生成机制对重入层进行编码,分别作为矩阵的第2列~第 L 列列向量。

3) 初始层-重入层加工系统的矩阵编码。

利用基于平行机机器号的工件加工机器流生成机制产生工件初始层和重入层加工机器流,如图3所示,最终形成二维矩阵形式。

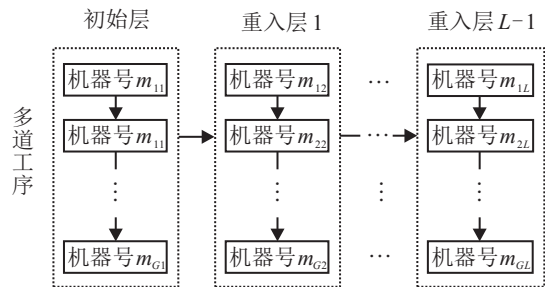


图3 初始层和重入层加工机器流

2.3 组批工件加工的元胞矩阵编码

J 个工件形成组批加工系统,组批加工系统内每个工件的加工机器流采用矩阵编码,从而形成由 J 个矩阵组成的元胞数组。一个元胞数组对应一条染色体,一个矩阵对应一个基因位值,利用自然数形式对每个染色体的基因位依次进行编号,工件在染色体中所处的位置即为该工件初始加工顺序,最终构成调度方案的元胞二维矩阵组形式,如图4所示。

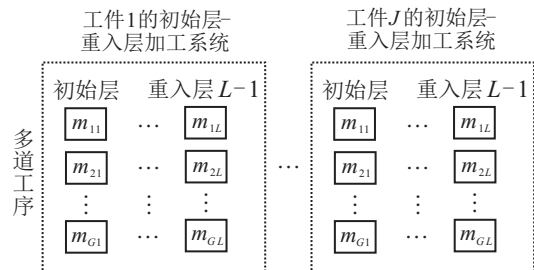


图4 组批工件加工系统

3 NEH-IGA融合求解算法

RHFS-TWC问题属于NP-hard问题,利用传统求解算法难以处理。NEH算法是用于解决置换流水车

间调度问题的启发式算法,其构造性极强.文献[15]证明了NEH算法可以解决可重入车间调度问题,但由于其是非迭代算法,求解效率不高.因此,本文根据可重入系统的特殊性,采用基于并行机机器号的二维矩阵编码解决工件在各道工序的机器分配,引入NEH规则产生工件初始加工顺序,从而形成RHFS-TWC问题的初始调度解.为避免算法陷入早熟,给出遗传参数的自适应动态调整机制进行解的更新,最终形成NEH-IGA融合求解算法.

3.1 基于NEH规则的工件初始加工顺序安排

调度解的二维矩阵编码解决了工件在各道工序的机器分配,但并未安排工件初始加工顺序,本文引入NEH算法予以解决,其操作步骤如下.

Step 1: 计算工件在所有机器上的总加工时间之和,进行降序排列;

Step 2: 选取总加工时间最大的两个工件,计算出能使总加权完成时间最小的排列作为工件加工序列;

Step 3: 将Step 1中产生的第 j 个工件,插入到工件加工序列的第 j 个位置,记录能使总加权完成时间最小的插入位置;

Step 4: 继续执行Step 3,直至Step 1中的所有工件全部插入,得到能使总加权完成时间较小的工件加工序列;

Step 5: 将Step 4产生的工件加工序列,对应于第2.3节产生的由 J 个二维矩阵组成的元胞数组,进行染色体基因位值的调整,得到可行且质量较高的初始调度解.

重复执行此操作,完成初始调度种群构建.

3.2 基于二维矩阵基因位取值的初始解生成方法

将元胞数组二维矩阵基因位取值方法获得的解作为问题的初始解.为了更简洁地表述RHFS-TWC加工系统,将工件 j 初始层和重入层加工机器流记为 $\lambda = \{\lambda_j | j = 1, 2, \dots, J\}$,其中 λ_j 表示工件 j 在各层次各道工序的加工机器选择,有

$$\lambda_j = \begin{bmatrix} \text{初始层} & \text{重入层1} & \dots & \text{重入层} L-1 \\ m_{11}^j & m_{12}^j & \dots & m_{1L}^j \\ \downarrow & \downarrow & \dots & \downarrow \\ m_{21}^j & m_{22}^j & \dots & m_{2L}^j \\ \downarrow & \downarrow & \dots & \downarrow \\ \vdots & \vdots & \vdots & \vdots \\ \downarrow & \downarrow & \dots & \downarrow \\ m_{G1}^j & m_{G2}^j & \dots & m_{GL}^j \end{bmatrix}.$$

Step 1: 基于平行机机器号二维矩阵编码的初始层基因位取值.

该基因位取值为解码的第1步,为元胞数组中各矩阵的第1列列向量,表示在初始层($l = 1$)第 g 道工序工件 j 在机器号为 m_{g1}^j 的机器上进行加工.显然所有工件机器流径路上的机器号应满足该工序的机器能力约束,即 $1 \leq m_{g1}^j \leq M_g$.

Step 2: 基于平行机机器号二维矩阵编码的重入层基因位取值.

该基因位取值为解编码的第2步,为元胞数组中各矩阵的第2列~第 L 列向量,表示在重入层($l = 1$)第 g 道工序工件 j 在机器号为 m_{gl}^j 的机器上进行加工.显然所有工件机器流径路上的机器号应满足该工序的机器能力约束,即 $1 \leq m_{gl}^j \leq M_g$.

为避免出现某工序某台机器分配较多工件,而某些机器相对空闲,令 m_{gl} 服从 $[1, M_g]$ 之间的均匀分布.

3.3 基于调度解的IGA更新过程

1) 基于适应度函数的调度解选择策略.

定义第 n 个调度解的适应度计算公式为

$$f_n = \frac{1}{\sum_{j=1}^J w_j C_{jGL}}. \quad (8)$$

选择策略应用轮盘赌规则,对于目标值较小的调度方案给予较大的选择概率.

2) 基于两点倒置交叉的调度解交叉机制.

为扩大解的搜索范围,实现工件加工顺序和机器分配的同步调整,采用两点倒置交叉规则.初始种群中所有个体在进行机器分配时,是从各工序既定机器集合中挑选一个机器编号,因此保证了个体的可行性.对于工件加工顺序,工件在染色体中所处的位置即为该工件初始加工顺序,执行两点之间的基因段倒置交换之后不会产生不可行后代,且能够同时拓展工件加工顺序和机器分配方案的搜索空间,故采用两点倒置交叉规则,步骤如下.

Step 1: 从上一代解群体中随机选择两个解 H_1 和 H_2 ;

Step 2: 工件顺序与加工机器流同步交叉过程,是在Step 1选择的解 H_1 和 H_2 中随机选择两个基因位 λ_a 和 λ_b ,将 λ_a 与 λ_b 之间的基因位进行翻转交换,产生新的子代染色体 O_1 和 O_2 ,其中 $a < b$, $a, b \in \bar{J}$,如图5所示.

3) 基于单点操作的调度解变异过程.

采用单点变异操作,在全局搜索中,选择某条染色体(二维矩阵组)的单个基因值 λ_j (二维矩阵),即某工件的加工机器流径路.为使工件初始层和重入层

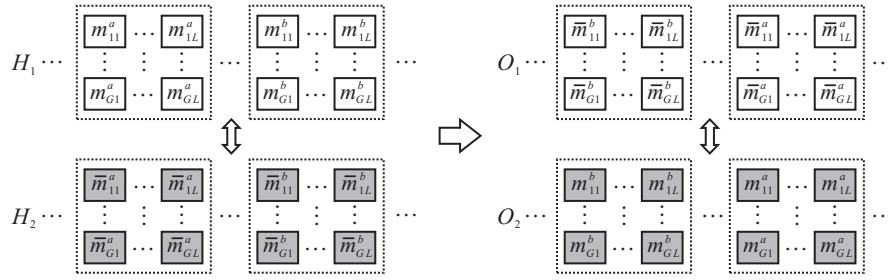


图5 调度解的两点倒置交叉

调度均发生变异,改变该工件某道工序所有层次的机器号,使之转化为近似解再进行择优,如图6所示。

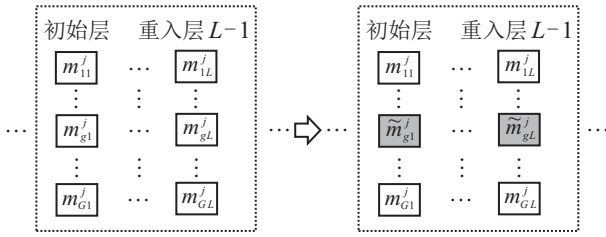


图6 单点基因位值变异

3.4 IGA 参数的自适应调整过程

设计第 i 次迭代的交叉概率 P_i 和变异概率 $\bar{P}_i$,使其随调度解群迭代次数和个体适应度的变化而自动调整。当个体适应度值较低时,对其执行较高的交叉和变异概率。当个体适应度值较高时,对其执行较低的交叉和变异概率^[16]。 P_i 与 $\bar{P}_i$ 的值设置如下。

Step 1: 计算调度解适应值。利用调度解适应函数计算公式(8)计算调度解群体中所有调度解的适应值,记为 $\{f_n | n = 1, 2, \dots, N\}$, N 为调度解群体规模,即调度解的个数。

Step 2: 计算调度解平均适应值。利用式 $f_{\text{avg}} = \sum_{n=1}^N \frac{f_n}{N}$ 计算调度解群体中所有调度解的适应值平均值,称为平均适应度,记为 f_{avg} 。

Step 3: 计算调度解最大适应值。从调度解群体中选择适应值最大的调度解,记为 f_{max} 。

Step 4: 设置交叉概率上下界区间。

在迭代初期,为扩大调度解群体搜索范围,设置较大的交叉概率取值,从而使选中的调度解有较大交叉可能性;在迭代中后期,为保留调度解群体中的优质调度解,但又确保调度解群体存在进一步的改进空间,采取缩小交叉概率的办法。基于上述原则,这里采取3段渐变式经验设置方法。

1) 迭代初期。设定迭代次数 i 低于总迭代次数 I 四分之一时为迭代初期,即 $i \leq I/4$ 。为扩大解群体搜索范围,设置较大的交叉概率取值,这里采取经验值设置方法, $[P_{\min}, P_{\max}] = [0.3, 0.9]$,从而使选中的解

具有较大交叉可能性。

2) 迭代中期。设定迭代次数 i 高于总迭代次数 I 四分之一但低于四分之三时为迭代中期,即 $I/4 < i \leq 3I/4$ 。采取大小适中的交叉概率,这里采取经验值设置方法, $[P_{\min}, P_{\max}] = [0.3, 0.7]$ 。

3) 迭代后期。设定迭代次数 i 高于总迭代次数 I 四分之三时为迭代后期,即 $i > 3I/4$ 。为保留解群体中的优质解,但又确保解群体存在进一步的改进空间,采取缩小交叉概率的办法,这里采取经验值设置方法, $[P_{\min}, P_{\max}] = [0.3, 0.5]$,从而使选中的解具有至多达到一半的交叉可能性。

Step 5: 确定动态交叉概率。

1) 对于适应值 f_n 大于平均适应度 f_{avg} 的调度解,为保证适应值较大的调度解具有较小的交叉概率,且随着迭代次数增加,交叉概率呈现逐步递减趋势,故设计受适应值和迭代次数双重影响的动态交叉概率计算方法,即

$$P_i = \begin{cases} P_{\max} - (P_{\max} - P_{\min}) \left(\frac{i}{I} + \frac{f_n}{f_{\max} - f_{\text{avg}}} \right), & f_n \geq f_{\text{avg}}; \\ P_{\max}, & f_n < f_{\text{avg}}. \end{cases} \quad (9)$$

2) 对于适应值 f_n 小于平均适应度 f_{avg} 的调度解,不考虑迭代次数影响,均按照交叉概率上界限值安排,即 $P_i = P_{\max}$ 。

Step 6: 设置变异概率上下界区间。

在迭代初期,为保留优质调度解,此时设置较小的变异概率取值,从而确保选中的调度解有合适的变异可能性;在迭代后期,调度解群体之间具有相似的基因结构,为避免过早收敛,采取增大变异概率的办法,从而确保选中的调度解具有较大变异的可能性。基于上述原则,这里采取3段渐变式经验设置方法。

1) 在迭代初期,为保留优质解的基因,此时设置较小的变异概率,采取经验值设置方法, $[\bar{P}_{\min}, \bar{P}_{\max}] = [0.01, 0.2]$,从而确保选中的解个体有合适的

变异可能性.

2) 在迭代中期, 进一步增大变异概率, 这里采取经验值设置方法, $[\bar{P}_{\min}, \bar{P}_{\max}] = [0.05, 0.2]$.

3) 在迭代后期, 解群体之间具有相似的基因结构, 为避免过早收敛, 采取增大变异概率的办法, 同样采取经验值设置方法, $[\bar{P}_{\min}, \bar{P}_{\max}] = [0.1, 0.2]$, 从而确保选中的解具有较大变异的可能性.

Step 7: 确定动态变异概率.

1) 对于适应值 f_n 大于平均适应度 f_{avg} 的解, 为使优质调度解可以转化为相似解再进行择优, 对于适应值较大的调度解设置较大的变异概率, 且随着迭代次数增加, 变异概率呈现逐步递增趋势, 故设计随适应值和迭代次数双重影响的变异概率计算方法, 即

$$P_i = \begin{cases} \bar{P}_{\max} - (\bar{P}_{\max} - \bar{P}_{\min}) \left(\frac{i}{I} + \frac{f_n}{f_{\max} - f_{\text{avg}}} \right), & f_n \geq f_{\text{avg}} \\ \bar{P}_{\min}, & f_n < f_{\text{avg}}. \end{cases} \quad (10)$$

2) 对于适应值 f_n 小于平均适应度 f_{avg} 的调度解不考虑迭代次数影响, 均按照变异概率下界限值安排, 即 $\bar{P}_i = \bar{P}_{\min}$.

可调参数 i 、 $P_{\min}$ 、 $P_{\max}$ 、 $\bar{P}_{\min}$ 、 $\bar{P}_{\max}$ 的选取对算法性能的改善程度和速度有较大影响. 有研究表明, 对于不同的应用对象并无通用的取值法则, 但对于同一类型问题, 能够找到一组较佳组合. 因此, 本文首先采用经验取值法估计取值区间, 然后通过大量的仿真实验测试对比, 结合问题特点, 深入分析后选取一组较佳组合.

4 实验测试与分析

4.1 实验设计

鉴于实际生产环境中时间的重要性以及所查阅的相关文献的优化方法, 为验证所提出算法的性能, 对传统 GA、仅引入遗传算法自适应调整策略的改进遗传算法 (IGA)、NEH 启发式以及 NEH-IGA 算法进行仿真实验. 随机产生多组数据, 分别进行如下测试.

1) 初始调度解群的质量对比实验. 实验对比了利用 NEH 算法产生初始调度解群和随机产生初始调度解群的质量, 从而分析引入 NEH 算法对 IGA 的影响.

2) 遗传参数自适应调整的效果实验. 为客观分析引入自适应交叉和变异概率对算法的影响, 进行基于固定交叉、变异概率的 GA 和基于自适应调整交叉、变异概率的 IGA 算法对比实验.

3) 不同规模 RHFS-TWC 实验测试. 为进行算法

的全方位对比分析, 设计不同规模问题的仿真实验, 对不同算法的求解效果进行比较.

通过 Microsoft Visual C++ 编程, 在 AMD A4-3300 M CPU 1.9 GHz 微机上运行. GA、IGA 和 NEH-IGA 融合算法的种群规模均设置为 80, 采用最大遗传代数 120 及运行时间 500 s 作为停止条件. 工件释放时间、工序间运输时间、工件加工时间、权重分别满足 $[1, 5]$ 、 $[1, 6]$ 、 $[1, 9]$ 、 $[1, 10]$ 之间的均匀分布.

4.2 初始调度解群的质量对比实验

为分析引入 NEH 算法产生初始种群对解的质量的影响, 实验对比了 NEH-IGA 和 IGA 算法的初始种群解的质量. 当 $J = \{30, 60, 80, 100\}$, $G = 3$, $L = 2$, 每道工序均有两台平行机时, 对每组实例运行 10 次取平均值, 实验结果如表 1 所示.

表 1 IGA 算法和 NEH-IGA 算法的结果对比

$J \times M \times G \times L$	IGA		NEH-IGA	
	初始种群 最优解	初始种群 均值	初始种群 最优解	初始种群 均值
$30 \times 2 \times 3 \times 2$	26 701	26 932.3	25 934	26 169.6
$60 \times 2 \times 3 \times 2$	94 932	95 228.8	93 528	93 710.3
$80 \times 2 \times 3 \times 2$	146 928	147 270.2	143 467	144 457.1
$100 \times 2 \times 3 \times 2$	227 920	229 801.3	220 187	223 704.9
平均值	124 120.3	124 808.2	120 779.0	122 010.5

由表 1 可看出, 对于不同规模的问题, NEH-IGA 的初始调度解群最优解的平均值为 120 779, 初始调度解群均值的平均值为 122 010.5; IGA 的初始调度解群最优解的平均值为 124 120.3, 初始调度解群均值的平均值为 124 808.2. 由此说明, 引入 NEH 算法提高了初始调度解的质量.

4.3 遗传参数自适应调整的效果实验

为公平比较算法, 传统 GA 的交叉概率 P_c 取固定值 0.7, 变异概率取固定值 0.05, 设置 GA 的停止时间与 IGA 进行 120 代的运行时间相同, 对 4.2 节的每种工件规模各取一个实例, 绘制出 GA 和 IGA 求解实例的目标值变化趋势, 如图 7 所示.

由图 7 可看出: GA 收敛较早, 容易陷入局部最优; IGA 收敛较晚, 在迭代后期能够扩大搜索范围, 以获得较好的解. 由此说明, 引入自适应交叉和变异概率改善了解的质量.

4.4 不同规模 RHFS-TWC 实验测试

对于不同规模的 RHFS-TWC 问题, 分别利用 NEH、IGA 和 NEH-IGA 进行仿真实验. M_g 、 L 、 G 、 J 分别取为 $\{2, 3\}$ 、 $\{2, 3\}$ 、 $\{3, 4, 5\}$ 、 $\{30, 60, 80, 100\}$, 假设每道工序的平行机数相同, 但所提出算法也适用于

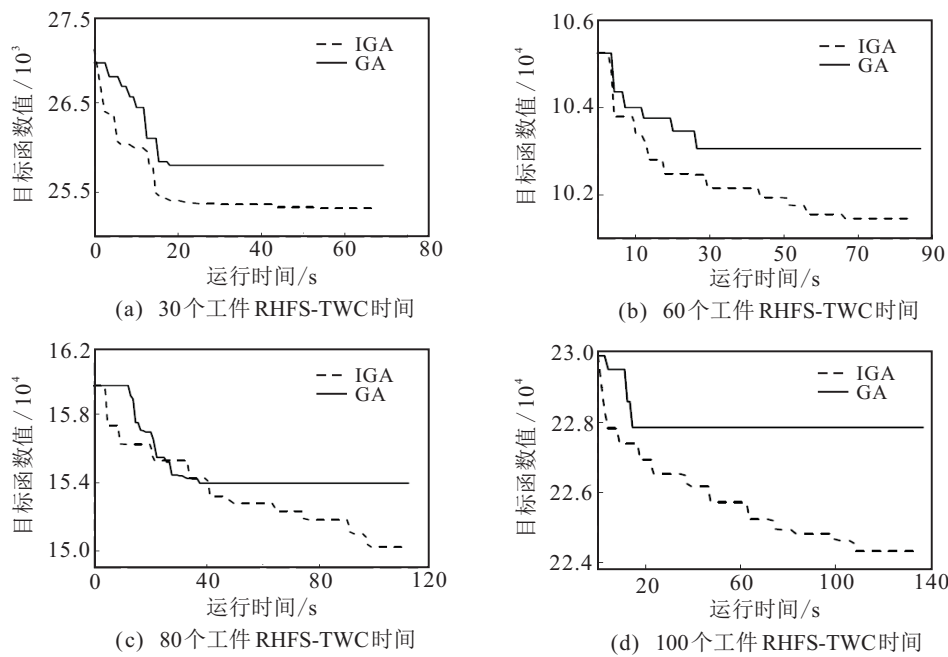


图7 不同规模RHFS-TWC问题的目标值变化趋势

表2 中小规模RHFS-TWC的测试结果

$J \times M \times G \times L$	min O			CPU时间/s		
	NEH	IGA	NEH-IGA	NEH	IGA	NEH-IGA
30×2×3×2	27 195	26 082	25 248	1.07	40.84	43.04
30×3×3×2	20 602	19 836	19 148	1.34	44.88	45.69
30×2×4×2	29 597	28 131	27 236	1.18	55.87	56.09
30×3×4×2	23 488	22 255	21 592	1.18	60.72	61.36
30×2×5×2	29 981	28 620	27 836	1.22	69.88	66.02
30×3×5×2	25 830	24 526	23 750	1.48	72.78	75.47
30×2×3×3	35 718	34 128	33 001	1.59	70.47	67.61
30×3×3×3	31 918	30 393	29 454	2.09	64.30	65.89
30×2×4×3	40 874	38 653	37 583	2.26	76.97	78.57
30×3×4×3	31 886	30 405	29 401	1.93	86.44	89.56
30×2×5×3	44 937	42 923	41 572	2.45	101.50	109.93
30×3×5×3	37 480	35 537	34 391	1.87	104.98	106.04
60×2×3×2	97 289	94 454	91 142	2.16	60.35	67.73
60×3×3×2	70 003	68 299	65 201	2.00	68.34	66.28
60×2×4×2	95 075	92 845	88 821	2.60	85.01	86.23
60×3×4×2	72 317	69 721	67 190	2.27	89.90	90.12
60×2×5×2	107 388	102 353	98 836	2.33	106.96	108.76
60×3×5×2	78 766	74 988	72 759	2.41	118.28	116.93
60×2×3×3	148 188	144 157	139 941	2.26	97.68	99.22
60×3×3×3	112 084	108 011	103 969	2.01	99.13	100.24
60×2×4×3	148 972	142 884	139 196	2.07	122.07	129.47
60×3×4×3	105 411	100 497	97 255	1.92	134.09	130.30
60×2×5×3	157 681	148 003	143 852	1.90	157.93	161.37
60×3×5×3	112 115	107 197	103 376	2.35	173.15	175.99
平均值	70 200	67 287	65 073	1.91	90.10	91.58

不同工序有不同机器数的问题. M_g 、 L 、 G 、 J 的不同取值共产生 $2 \times 2 \times 3 \times 4 = 48$ 种组合,每个组合随机产生并测试10组实例,因此本实验共测试了480组实例,实验结果如表2和表3所示.表2和表3中的值为相同规模问题的10个实例的平均值(除最后1行).实验结果通过目标改进率 α_1 和 α_2 来衡量,其中

$$\alpha_1 = \frac{O_{\text{NEH}} - O_{\text{NEH-IGA}}}{O_{\text{NEH-IGA}}} \times 100\%,$$

$$\alpha_2 = \frac{O_{\text{IGA}} - O_{\text{NEH-IGA}}}{O_{\text{NEH-IGA}}} \times 100\%.$$

可得到以下结论:

1) 对于不同规模的问题,3种算法都能够在合理的时间内得到较好的近优解. NEH算法的运行时间最短,总平均运行时间为2.19s,但求得的解较差. NEH-IGA算法的总平均运行时间为121.46s,但求解质量较NEH算法提高了8.8%. IGA算法的总平均运

表 3 大规模RHFS-TWC的测试结果

$J \times M \times G \times L$	min O			CPU时间/s		
	NEH	IGA	NEH-IGA	NEH	IGA	NEH-IGA
80×2×3×2	152 751	144 839	141 217	1.80	74.33	76.35
80×3×3×2	119 520	111 693	108 932	2.12	81.07	83.73
80×2×4×2	171 994	162 718	158 546	1.90	104.99	108.98
80×3×4×2	129 838	123 257	120 091	2.20	112.35	115.77
80×2×5×2	173 853	167 462	163 079	2.13	132.92	137.87
80×3×5×2	145 133	130 346	127 615	1.78	140.83	132.74
80×2×3×3	265 994	245 902	238 353	2.13	116.60	117.95
80×3×3×3	177 760	165 631	161 642	2.64	119.53	122.76
80×2×4×3	264 340	251 418	244 771	2.25	166.59	165.48
80×3×4×3	191 823	176 531	172 136	2.74	169.01	170.42
80×2×5×3	274 851	264 453	257 274	1.96	199.64	200.44
80×3×5×3	206 972	191 645	186 556	2.07	211.74	208.64
100×2×3×2	233 273	224 832	216 843	4.57	96.49	98.36
100×3×3×2	184 157	170 571	164 096	3.12	95.69	100.33
100×2×4×2	259 017	244 981	238 495	2.50	128.21	132.12
100×3×4×2	182 067	174 237	169 745	2.37	135.14	133.25
100×2×5×2	262 351	251 654	244 238	2.67	165.86	165.85
100×3×5×2	186 734	171 015	166 825	2.65	177.58	178.05
100×2×3×3	398 389	383 425	373 085	4.06	139.70	137.49
100×3×3×3	247 279	240 015	233 645	2.22	145.97	144.64
100×2×4×3	398 162	366 070	356 147	2.45	192.75	193.52
100×3×4×3	309 882	288 059	280 895	2.32	213.60	211.29
100×2×5×3	417 951	397 583	387 293	1.92	241.74	245.08
100×3×5×3	299 852	279 202	272 087	2.42	252.49	250.85
平均值	235 581	221 981	215 984	2.46	150.62	151.33

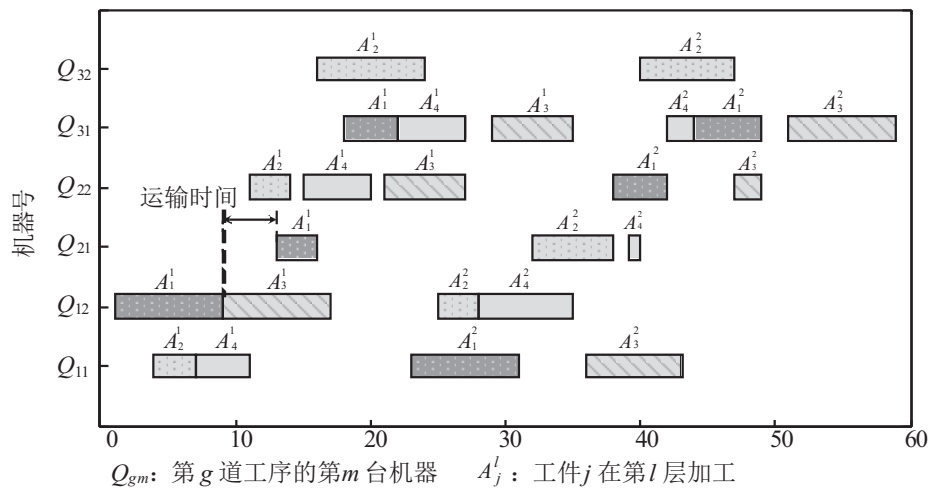


图 8 调度时间表

行时间为120.36 s,虽然NEH-IGA的运行时间比IGA略高,但解的质量提高了2.92 %.

2) 对于中小规模 RHFS 问题,IGA 在平均 90.1 s 内求得的近优解为 67 287,NEH-IGA 在平均 91.58 s 内求得的近优解为 65073. NEH-IGA 计算时间比IGA 高了 1.08 s,但求解质量较 IGA 提高了 3.4 %,较 NEH 算法提高了 7.88 %.

3) 对于大规模 RHFS 问题,IGA 在平均 150.62 s 内求得的近优解为 221 981,NEH-IGA 在平均 151.33 s 内求得的近优解为 215 984. NEH-IGA 的计算时间比 IGA 高了 0.71 s,但求解质量较 IGA 提高了 2.78 %,较 NEH 提高了 9.1 %. 与中小规模问题相比,实验结果的目标值改进率 α_1 提高了 1.22 %,说明随着问题规模的增加,与 NEH 算法相比,NEH-IGA 的优势更明显.

4) IGA、NEH-IGA 算法在产生初始调度解群后,为获得更好的调度解而进行自适应新解更新,每次迭代都需进行大量的计算和筛选,故其 CPU 运行时间较长,但调度解的质量与 NEH 算法相比均有较大提高,其平均性能均优于 NEH 算法.

4.5 RHFS-TWC的调度时间表

假设每道工序的平行机数相同, $\{M_g, L, G, J\}$ 为 $\{2, 2, 3, 4\}$, 工序间的运输时间 $Y_{l,g,g+1}$ 为 $\{1, 4, 2\}$, 权重为 $\{4, 5, 1, 6\}$, 释放时间为 $\{1, 4, 2, 3\}$, 各工件的加工时间设置如下所示(每行对应1个工件, 上标表示所处层):

$$T = \begin{bmatrix} 8' & 3' & 4' & 8'' & 4'' & 5'' \\ 3' & 3' & 8' & 3'' & 6'' & 7'' \\ 8' & 6' & 6' & 7'' & 2'' & 8'' \\ 4' & 5' & 5' & 7'' & 1'' & 2'' \end{bmatrix}.$$

利用上述实例中的数据, 采用NEH-IGA算法进行求解, 进而绘制调度时间表, 如图8所示.

5 结论

本文研究了在电子工业中应用广泛的RHFS-TWC问题, 建立了整数规划模型, 提出了NEH-IGA算法. 由于可重入生产系统的特殊性, 本文提出一种二维矩阵组编码方案, 从初始调度解群生成、交叉与变异概率的自适应动态调整两方面提高算法性能. 对大量随机生成的数据进行测试, 通过初始种群质量对比实验、遗传参数改进效果实验和不同规模RHFS-TWC仿真实验, 表明NEH启发式和遗传概率自适应动态调整策略的引入有效地改善了原有GA的求解能力. 与NEH算法相比, NEH-IGA算法得到的解的质量有较大提高, 但运行时间较长; 与IGA算法相比, 虽然运行时间略长, 但解的质量明显有较大改善. 未来的研究方向可考虑如何减少算法运行时间以及用所提出算法求解多目标RHFS-TWC问题.

参考文献(References)

- [1] Lin D, Lee C K M. A review of the research methodology for the re-entrant scheduling problem[J]. Int J of Production Research, 2010, 49(8): 2221-2242.
- [2] Ying K C, Lin S W, Wan S Y. Bi-objective reentrant hybrid flow shop scheduling: An iterated Pareto greedy algorithm[J]. Int J of Production Research, 2014, 52(19): 5735-5747.
- [3] Yang F, Xi W L. Online parallel-batch scheduling to minimize total weighted completion time on single unbounded machine[J]. Information Processing Letters, 2016, 116(8): 526-531.
- [4] Martin S. A 2.542-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective[J]. Operations Research Letters, 2016, 44(5): 676-679.
- [5] Chen J F. Unrelated parallel-machine scheduling to minimize total weighted completion time[J]. J of Intelligent Manufacturing, 2015, 26(6): 1099-1112.
- [6] Yuan H W, Jing Y W, Ren T. A heuristic algorithm for flow shop scheduling problem[C]. The 2nd Asian Pacific Conf on Mechatronics and Control Engineering. Hong Kong, 2014, 643: 374-379.
- [7] 轩华. 运输能力有限混合流水车间调度的改进拉格朗日松弛算法[J]. 计算机集成制造系统, 2013, 19(7): 1634-1639.
(Xuan H. Improved LR algorithm for hybrid flow shop scheduling with transportation consideration[J]. Computer Integrated Manufacturing Systems, 2013, 19(7): 1634-1639.)
- [8] 轩华, 孙振轩, 李冰. 零等待混合流水车间问题优化研究[J]. 工业工程与管理, 2014, 19(5): 13-17.
(Xuan H, Sun Z X, Li B. Research on optimization of zero-wait hybrid flowshop problem[J]. Industrial Engineering and Management, 2014, 19(5): 13-17.)
- [9] Chou F D, Wang H M. A simulated annealing to solve four-stage open shops with parallel machines[C]. The 2nd Int Conf on Materials Engineering and Automatic Control. Ji'nan, 2013, 330: 843-847.
- [10] Choi H S, Kim H W, Lee D H, et al. Scheduling algorithms for two-stage reentrant hybrid flow shops: Minimizing makespan under the maximum allowable due dates[J]. Int J of Advanced Manufacturing Technology, 2009, 42(9/10): 963-973.
- [11] Shen J N, Wang L, Zheng H Y. A modified teaching-learning-based optimization algorithm for bi-objective re-entrant hybrid flowshop scheduling[J]. Int J of Production Research, 2016, 54(12): 3622-3639.
- [12] Ying K C, Lin S W, Wan S Y. Bi-objective reentrant hybrid flowshop scheduling: an iterated Pareto greedy algorithm[J]. Int J of Production Research, 2014, 52(19): 5735-5747.
- [13] Dugardin F, Yalaoui F, Amodeo L. Hybrid multi-objective methods to solve reentrant shops[J]. Int J of Applied Logistics, 2012, 3(4): 15-32.
- [14] 周炳海, 钟臻怡. 可重入混合流水车间调度的拉格朗日松弛算法[J]. 控制理论与应用, 2015, 32(7): 881-886.
(Zhou B H, Zhong Z Y. Lagrangian relaxation algorithm for scheduling problems of reentrant hybrid flow shops[J]. Control Theory & Applications, 2015, 32(7): 881-886.)
- [15] Pan J C, Chen J S. Minimizing makespan in re-entrant permutation flow-shops[J]. J of Operational Research Society, 2003, 54(6): 642-653.
- [16] 雷伟军, 程筱胜, 戴宁, 等. 基于改进遗传算法的多模型加工路径规划[J]. 机械工程学报, 2014, 50(11): 153-161.
(Lei W J, Cheng X S, Dai N, et al. Multi-model machining path planning based on improved genetic algorithm[J]. J of Mechanical Engineering, 2014, 50(11): 153-161.)

(责任编辑: 孙艺红)