

带队列约束的 RHFS 列生成调度算法

周炳海, 王 科
(同济大学 机械与能源工程学院, 上海 201804)

摘 要: 为有效提升多重入车间的生产效率,考虑实际生产中队列约束,提出了基于列生成算法的可重入混合流水车间的调度方法. 首先对两阶段生产调度问题进行描述,以最小化工件总完成时间为优化目标,建立数学规划模型. 针对该调度模型提出列生成算法,设计带多重决策的动态规划方法来求解工件级子问题,为更快收敛,主问题求解中采用自适应加速策略. 在使用分支定界将得到的解整数化的过程中,构造列池并设计局部变异. 最后,对各种不同问题规模进行了数值实验,结果表明所提出的调度算法是有效可行的.

关 键 词: 队列;可重入;列生成;动态规划;分支定界

中图分类号: TP 391 **文献标志码:** A **文章编号:** 1005-3026(2018)09-1315-06

Column Generation Scheduling Algorithm of Reentrant Hybrid Flow Shops(RHFS) with Queue Constraints

ZHOU Bing-hai, WANG Ke
(School of Mechanical Engineering, Tongji University, Shanghai 201804, China. Corresponding author: ZHOU Bing-hai, E-mail: bhzhou@tongji.edu.cn)

Abstract: To effectively enhance the production efficiency of multi-reentrant workshop, the queue constraint was considered where products were processed layer by layer, and then a scheduling method of reentrant hybrid flow shops (RHFS) was proposed based on column generation algorithm. Firstly, a two-stage scheduling model of RHFS was described and a mathematical programming model was built with an objective of minimizing the total completion time. A column generation algorithm was developed and dynamic programming with multiple decision-making was designed to solve each sub-problem. Further, the adaptive accelerating strategy was applied to effectively improve the algorithm convergence. In the process of generating integral solutions by using branch-and-bound method, column pool was built and neighborhood mutation method was employed. Finally, numerical experiments in different problem scales were carried out to analyze the proposed algorithm. Results verify the validness and feasibility of the proposed algorithm.

Key words: queue; reentrant; column generation; dynamic programming; branch-and-bound

可重入车间作为半导体系统中最复杂的调度,其基本特性是一个工件多次进入某些工作站^[1-2]. 在半导体晶圆制造过程中,晶圆需多次经过相同的设备进行加工,重入加工的次数取决于晶圆的层数. 晶圆重入这一制造过程即为可重入混合流水车间(reentrant hybrid flow shops, RHFS)问题.

RHFS 自 1983 年被 Graves 提出以来^[3],已取得较大研究进展. Hekmatfar 等^[4]研究了只含两道工序,以最小 makespan 为优化目标提出了一种混合遗传算法. Dugardin 等^[5]以瓶颈利用率最大化和最大完成时间最小化为调度目标,提出一种新的 L-NSGA 算法. Cho 等^[6]提出了基于局部搜索的帕累托遗传算法求解 RHFS 调度问题. Jeong 等^[7]以最小化延迟时间为目标函数,提出了启发式算法和分支定界算法. Choi 等^[8]针对两

阶段 RHFS 问题,以最小化 makespan 为目标提出了分支定界和启发式算法.

上述文献能够为 RHFS 调度问题提供良好的借鉴,但未考虑实际生产中存在的队列约束^[9-10]. 队列约束是指工件完成某工序加工后,需在一定时间内到达下一个工作站进行加工,否则工件将会报废或返工^[11]. 其次上述文献多使用智能优化算法,未体现工件不同层次间对于机器耦合这一特性,且未使用列生成这类能够在合理时间内找到可量化指标的近优解算法. 因此本文将队列约束考虑进 RHFS 调度问题中,并使用列生成算法求解问题的近优解.

1 问题描述

为便于描述,对符号做如下定义. i, i' 为工件索引; l 为加工层索引; j 为工作站 1 上的机器索引; o 为批的索引; H 为批的总数; N 为工件的总数; L 为工件的层数; M 为工作站 1 上的机器总数; t_l 为工件的 l 层在专用机器上加工的时间; t'_l 为工件的 l 层在普通机器上加工的时间; $S_{i,l,2}$ 为工件 i 的 l 层在工作站 2 上开始加工的时间; $C_{l,i,1}$ 为工件 i 的 l 层在工作站 1 上的完成时间; $C_{l,i,2}$ 为工件 i 的 l 层在工作站 2 上的完成时间; C_o 表示批 o 的完成时间; w_l 为工件 l 层等待时间的阈值; $Z_{i,i',l}$ 为工件 i 及工件 i' 的 l 层在工作站 2 上的同一批中加工; $x_{l,i,j,k}$ 为工件 i 的 l 层在机器 j 的可行排序的第 k 个位置加工; $t_{l,2}$ 为工件的 l 层在工作站 2 的加工时间; $t_{l,1}$ 为工件 i 的 l 层在工作站 1 上的加工时间; $\varphi_{j,l,i}$ 表示工件 i 的 l 层在机器 j 上加工; B 为批容量的上限值.

本文所研究的 RHFS 调度问题中,由 2 个工作站组成, N 个工件需重复进入 2 个工作站加工 L 次,如图 1 所示. 工作站 1 由 M^1 台机器组成,每台机器同一时间只能加工 1 个工件; 工作站 2 由

1 台批处理的机器组成,机器同一时间只能加工一个批.

工件每一层都有专用的加工机器,工件在专用机器上加工的时间为普通机器上加工的时间的 μ 倍,如式(1)所示:

$$t_l = \mu t'_l, l = 1, 2, \dots, L. \tag{1}$$

工件在工作站 1 完成加工后,即进入队列等待在工作站 2 上进行加工. 为保证产品质量,工件在工作站 1 完成加工后至下一个工作站加工的等待时间不能超过规定的阈值,即需满足式(2):

$$S_{i,l,2} - C_{l,i,1} < w_l, i = 1, 2, \dots, N, l = 1, 2, \dots, L. \tag{2}$$

只有加工相同层的工件才能形成一个批,批的容量不能超过上限 B ,且同一批加工的工件具有相同的完成时间,即满足式(3),式(4). 本文将处理同一层上的相同工序定义为同一作业.

$$\sum_{i=1, i \neq i'}^N Z_{i,i',l} \leq B, i' = 1, 2, \dots, N, l = 1, 2, \dots, L; \tag{3}$$

$$C_{l,i,2} \geq Z_{i,i',l} \cdot C_{i',l,2}, i, i' = 1, 2, \dots, N, l = 1, 2, \dots, L. \tag{4}$$

定义 $\kappa_m (m = 1, 2, \dots, M)$ 为工作站 1 中机器 m 的可行排序,可行排序的每个位置表示需要加工的作业,其长度设为 $N \times L$. 任一位置至多分配一个作业,位置为空表示无作业分配,只有上一位置被分配后才能分配下一个位置,即需满足式(5)~式(7).

$$\sum_{l=1}^L \sum_{i=1}^N x_{l,i,j,k} \geq \sum_{l=1}^L \sum_{i=1}^N x_{l,i,j,k+1}, \tag{5}$$
$$j = 1, 2, \dots, M, k = 1, 2, \dots, N \times L; \tag{5}$$

$$\sum_{l=1}^L \sum_{i=1}^N x_{l,i,j,k} \leq 1, j = 1, 2, \dots, M, k = 1, 2, \dots, N \times L; \tag{6}$$

$$\sum_{l=1}^L \sum_{i=1}^N (C_{l,i,1} + t_{l,i,1}) \cdot x_{l,i,j,k} \leq \sum_{l=1}^L \sum_{i=1}^N C_{l,i,1} \cdot x_{l,i,j,k+1}, j = 1, 2, \dots, M, k = 1, 2, \dots, N. \tag{7}$$

作业只能安排在一个可行排序的一个位置上,即需满足式(8).

$$\sum_{j=1}^M \sum_{i=1}^{N \times L} x_{l,i,j,k} = 1, i = 1, 2, \dots, N, l = 1, 2, \dots, L. \tag{8}$$

任一工件,只有在前一层加工完成后,后一层才能开始加工,需满足式(9):

$$C_{l-1,i,2} + t_{l,i,1} \leq C_{l,i,1}, i = 1, 2, \dots, N, l = 1, 2, \dots, L. \tag{9}$$

任一工件,只有在工作站 1 完成加工后,才能

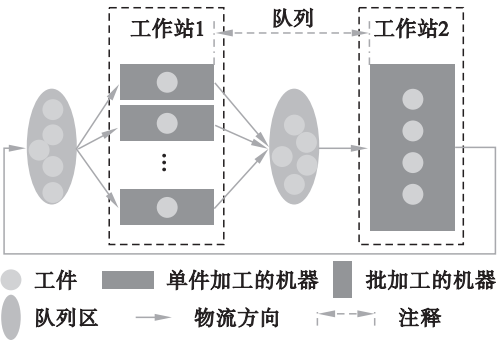


图 1 考虑队列约束的 RHFS 问题

Fig. 1 Problem of RHFS with queue constraints

到工作站 2 进行加工,需满足式(10):

$$C_{l,i,1} + t_{l,2} \leq C_{l,i,2}, i = 1, 2, \dots, N, l = 1, 2, \dots, L. \quad (10)$$

调度目标为所有工件的完成时间之和最小化,如式(11)所示:

$$\min \sum_{i=1}^N C_{L,i,2}. \quad (11)$$

2 列生成算法

为使用列生成方法进行求解,需要把上述模型建成基于时间-设备二维列的集合分割模型的形式.

2.1 主问题 SP 模型的建立

为方便运用列生成,做如下定义: s 为单个工件在各台机器上的调度; t 为时间刻度; T 为满足调度需要的总时间长度; $f_s = C_s$ 为调度 s 的成本; y_s 表示某一工件按照调度 s 加工,则为 1, 否则为 0; a_{jts} 表示调度 s 于时间 t 在工作站 1 的机器 j 上加工,则为 1, 否则为 0; b_{os} 表示调度 s 分配到批 o 加工则为 1, 否则为 0;通过 D-W 方法,上述问题可以分解为集合分割模型的主问题和子问题^[12].

$$\text{主问题: } \min \sum_{s \in \Omega} f_s y_s.$$

s. t.

$$\sum_{s \in \Omega} \sum_{l \in L} a_{jts} y_s = 1, t = 1, 2, \dots, T, j = 1, 2, \dots, M; \quad (12)$$

$$\sum_{s \in \Omega} b_{os} y_s \leq B, o = 1, 2, \dots, H; \quad (13)$$

$$\sum_{s \in \Omega} y_s = N; \quad (14)$$

$$y_s \in \{0, 1\}, \forall s \in \Omega. \quad (15)$$

上述 SP 模型中, y_s 的取值为 0 或 1. 将约束(15)修改为 $y_s \geq 0$ 时,得到原问题的线性松弛问题 LSP,求解 LSP 问题,将得到原问题的下界.

构造列时需将可行调度 s 中机器和时间的信息集成到一列中. 每一列由 $[a_{jts}, b_{os}, 1]^T$ 构成,表示一个工件可行调度.

Ω 表示符合条件的工件可行调度的集合. 运用列生成算法求解 LSP 问题时,只需给定部分可行的列 $\Omega' \subseteq \Omega$,即求解限制性松弛问题(RLMP).

记 $u_{jt}(t = 1, 2, \dots, T, j = 1, 2, \dots, M)$ 为对应于等式约束(12)的对偶变量值, $\delta_o(o = 1, 2, \dots, H)$ 为对应于约束(13)的对偶变量值, v 为对应于约束(14)的对偶变量值. 因为所有的工件都相同,只需解决一个低级子问题.

$$\min RC_s = f_s - \sum_{t \in T} \sum_{j \in J} a_{jts} u_{jt} - \sum_{o \in H} \delta_o b_{os} - v. \quad (16)$$

若存在调度 s ,使 $RC_s < 0$,则说明主问题还未达到最优解. 通常需将 RC_s 值最小的列添加到 RLMP 中. 本文对列生成算法进行改进,采用自适应加速策略.

对于调度 s ,若满足 $RC_s < 0$ 且 $RC_s > \min RC$,则以一定概率被接收作为主问题的列. 由于前期 RLMP 中的列相对较少,算法注重宽度搜索,后期列较多,为避免计算冗杂,被接收的概率下降. 因此设置接收模型为

$$R_s = \begin{cases} 1, & RC_s = \min RC \text{ or } RC_s < 0 \cap \text{Rand}_s > \text{rnd}, \\ 0, & \text{otherwise}, \end{cases}$$

$$\text{rnd} = 1 - \sin\left(\frac{1}{G+1}\pi\right).$$

当调度 s 被接收作为主问题的列, R_s 为 1, 否则为 0; Rand_s 为调度 s 产生的介于 0 与 1 之间的随机数; rnd 为自适应算子; G 为当前的迭代次数.

2.2 求解工件级子问题

工件级子问题即寻找使得式(16)最小的列. 工件级子问题为

$$\min \{ C_{L,i,2} - \sum_{l=1}^L \sum_{t=C_{l,i,1}-t_{l,i,j,1}+1}^{C_{l,i,1}} \varphi_{j,l,i} u_{j,t} - \sum_{o=1}^{\omega} \delta_o b_{oi} \}. \quad (17)$$

s. t.

$$C_{l-1,i,2} + t_{l,i,1} \leq C_{l,i,1}, l = 1, 2, \dots, L; \quad (18)$$

$$C_{l,i,1} + t_{l,2} \leq C_{l,i,2}, l = 1, 2, \dots, L; \quad (19)$$

$$C_{l,i,1} + t_{l,2} + w_l \geq C_{l,i,2}, l = 1, 2, \dots, L; \quad (20)$$

$$\sum_{j=1}^M \varphi_{j,l,i} = 1, l = 1, 2, \dots, L; \quad (21)$$

$$\sum_{o=1}^{\omega} b_{oli} = 1, l = 1, 2, \dots, L; \quad (22)$$

$$C_{l,i,2} \geq b_{oli} \cdot C_o, l = 1, 2, \dots, L; \quad (23)$$

$$t_{l,i,1} \geq \varphi_{j,l,i} \cdot t_{l,j,1}, l = 1, 2, \dots, L, j = 1, 2, \dots, M. \quad (24)$$

工件级子问题就是以式(17)为目标函数,约束(18)~(20)表示工件需要满足的时间约束;约束(21)表示工件的每一层在工作站 1 上加工时只能在 1 台机器上加工;约束(22)表示工件任一层在工作站 2 上进行加工时选择的批的约束;约束(23)表示工件在工作站 2 上完成的加工的时间与该批完成的时间相同;式(24)表示工件在工作站 1 上的加工时间的约束.

2.3 工件级子问题动态规划算法

按工件层数划分阶段,阶段变量取 $l = 1, 2,$

..., L . 每个阶段的状态变量集合与其工作站 2 上加工的批相对应, 即 $X_l = \{o \mid b_{oli} = 1\}, l = 1, 2, \dots, L$.

定义工件 i 的第 l 层在工作站 1 上的机器 j 在时间 t 完成加工的阶段指标为 χ_{ijl} . 定义 Γ 为满足调度时间要求的最小时间刻度.

$$\chi_{ijl} = - \sum_{x=t-p_{l,i,j}+1}^t u_{j,x}, \forall j \in \{0, 1, 2, \dots, M\}, \forall t \in \Gamma.$$

定义工件 i 的第 l 层在工作站 2 的批 o 上进行加工的指标为 ξ_{ilo} :

$$\xi_{ilo} = \begin{cases} \min_{t \in T_o, \forall j \in J} (\chi_{ijl}) - \delta_o + C_o, & \text{if } l = L; \\ \min_{t \in T_o, \forall j \in J} (\chi_{ijl}) - \delta_o, & \text{otherwise;} \\ \forall o \in \{0, 1, 2, \dots, w\}. \end{cases}$$

其中 $T_o = \{t \mid C_o - t_{l,2} - w_l \leq t < C_o - t_{l,2} + 1\}$.

建立递归方程如下:

$$f_{il}(X_l, o) = \xi_{ilo} + \min_{\forall o' \in X_{l-1}'} f_{i(l-1)}(X_{l-1}, o').$$

其中: $X_{l-1}' = \{o' \mid b_{o'(l-1)i} = 1 \text{ 且 } C_{o'} < C_o\}$; $f_{il}(X_l, o)$ 表示工件 i 的 l 层采用策略 o 达到状态为 X_{l-1} 的最优值.

工件 i 在加工 l 层时获得的最优状态通过回溯法获得.

$$(X_l^*, o^*) = \arg \min_{\forall o \in X_L} f_{il}(X_l, o),$$
$$(X_l'^*, o'^*) = \arg \min_{\forall o' \in X_l' \cap \{o' \mid C_{o'} < C_{o^*}\}} f_{il}(X_l', o').$$

工件 i 在加工 l 层时在工作站 1 所选择最优的机器 j^* 与时间 t^* 为 $(j^*, t^*) = \arg \xi_{ilo}$.

2.4 分支定界整数化最优解

上面过程得到的解往往为非整数的解, 而实际调度问题中, 往往需要整数解. 因此就需要设计分支定界的规则使最终得到的解为整数^[13].

在分支的过程中, 若存在多个非整数解, 选择分支的解需满足 $\min\{|y_s - 0.5|\}$. 通过将 y_s 进行分支, 把原问题分成两个子问题(即对应 $y_s = 1$ 和 $y_s = 0$ 两个节点), 针对这两个子问题需做如下调整.

1) 限制性主问题的初始列, $y_s = 1$ 的分支在产生初始列时不考虑分配给 y_s 的任意时刻, 式(13)中分配给 y_s 的批相应减 1, 式(14)右边的 N 值相应减 1; $y_s = 0$ 在产生初始列时不产生与 y_s 一致的列.

2) 求解子问题时, $y_s = 1$ 的分支在新增列时不考虑已经分配给 y_s 的任意时刻; $y_s = 0$ 分支在产生新列时不产生与 y_s 一致的列. 由于很难保证不产生与 y_s 一致的列, 因此本文使用列池来存放需要避免的列. 当求解子问题得到的列与列池中

的列相同时, 则将该列进行局部变异. 变异分两种形式: 1) 对机器上的处理时间进行变异; 2) 对批进行变异. 对机器上的处理时间进行变异时, 将工件的处理时间往邻近的区域偏移一个时间单位; 对批进行变异时, 即改变工件所分配的批, 同时需要根据式(18)~式(20)调整工件在机器上的处理时间.

3 数值实验

为验证列生成算法的有效性, 设计具体算例进行求解. 第一阶段的机器数量 $m = 3$, 工件数量 $n = 10$, 工件层数 $l = 3$, 每个工件不同层在各机器的加工时间见表 1. 不同层组成的批在第二阶段的加工时间 $t_{l,2} = \text{rand}(3, 5), l = 1, 2, 3$, 批的上限值 B 为 3, 工件在第二阶段的等待时间 $w_l = 5, l = 1, 2, 3$.

表 1 不同层在各机器上的加工时间
Table 1 Processing time of different layer on each machine

机器 编号	工件层		
	1	2	3
1	2	4	5
2	3	3	5
3	3	4	4

通过 MATLAB 语言进行编程后, 在英特尔酷睿 2.5 GHz 处理器上运行, 得到总完成时间变化见图 2.

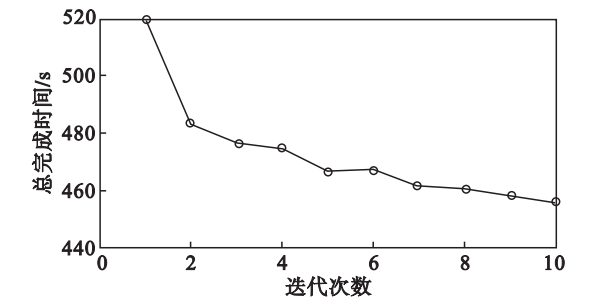


图 2 求解过程中总完成时间的变化
Fig. 2 Variation of total completion time during solving process

当检验数非正时, 得到该算例的 LSP 问题的最优解, 表明本文设计的列生成算法可以有效求解 LSP 问题.

用列生成算法对随机产生的实例进行了仿真实验. 分别考虑工作站 1 配置 3~6 台机器环境下, 对任意工件个数和机器台数的组合, 随机产生

10 组不同的实例数据,结果取平均值. 将列生成算法得到解和分支定界运行后得到的解分别与启发式给出的初始解做比,如表 2 所示.

从表 2 可以看出,本文算法对中等规模的问题是有效的,在可接受的时间范围能进行求解.但随着工件个数的增加,算法所产生的列和计算时间会急剧增加.列生成算法的列数和计算时间主要受工件个数的影响,受机器数量的影响不明显.

表 2 不同机器及工件规模下的列生成算法数值结果

Table 2 Numerical results of column generation algorithm under different sizes of machines and jobs

机器台数	工件个数	列生成算法的列数	算法运行时间/s	列生成算法求值	分支定界取整后
3	10	20	3.64	0.876 9	0.908 5
	30	577	68.73	0.803 3	0.844 9
	50	1 762	766.32	0.767 2	0.822 9
4	30	634	92.41	0.814 7	0.854 1
	50	1 892	1 189.36	0.870 3	0.881 1
	70	2 538	2 664.39	0.830 7	0.863 1
5	30	539	155.25	0.913 4	0.922
	50	1 862	1 367.8	0.721 9	0.737 6
	70	2 661	2 691.52	0.896 5	0.927
6	50	1 764	809.84	0.669 7	0.695 2
	70	2 713	2 689.78	0.820 7	0.838 1
	90	3 558	6 058.46	0.822 6	0.835 6

本文在 RHFS 调度中考虑了队列约束,因此对不同的问题分析了队列约束的影响.对机器台

数为 3 台(M3),工件层数为 3 层(L3),工件个数为 20,30,40(J20/J30/J40)的情况下进行了仿真.将运行的结果与不考虑队列约束情况下得到的结果做对比,如图 3 所示.

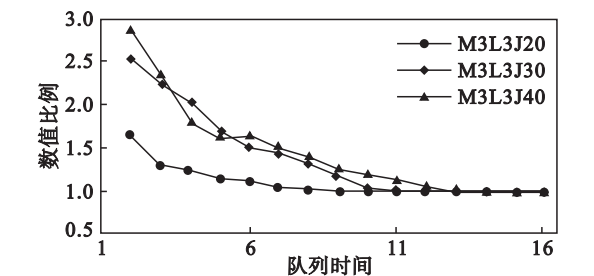


图 3 队列约束对 RHFS 调度的影响
Fig. 3 Influence of queue constraints on RHFS scheduling

由图 3 可知,队列约束对 RHFS 调度具有比较明显的影响,随着队列时间的增大,队列约束的影响逐渐减少,并在一定范围之后,队列约束对调度不再影响.

对于 RHFS 调度问题的求解,大多文献使用遗传算法(GA)或改进的 GA 对调度问题进行求解.通过将 GA 和列生成算法(CG)对多个 RHFS 调度问题进行求解,从调度目标总完成时间(TC)及算法的运行时间两个维度进行对比,如表 3 所示.

从表 3 可以发现,遗传算法可以在小规模问题的求解上得到近优解,但随着问题规模的增大,其与最优解之间的差距(gap)也逐渐增大;同时在求解大规模问题上易于陷入局部收敛,导致无法得到全局较优的解.列生成算法较 GA 算法有明显的优势.

表 3 不同规模 RHFS 调度问题下遗传算法和列生成算法求解的对比

Table 3 Contrast of genetic algorithm and column generation algorithm in RHFS scheduling problem with different size

算例	GA		CG		gap/%	算例	GA		CG		gap/%
	TC	t/s	TC	t/s			TC	t/s	TC	t/s	
M3L3J10	457	7.1	457	3.6	0.0	M3L4J10	742	19.8	742	5.3	0.0
M3L3J30	4 174	29.4	3 950	68.7	5.7	M3L4J30	8 038	42.4	7 921	77.3	1.5
M3L3J50	16 218	67.2	12 338	766.3	31.4	M3L4J50	34 462	102.6	30 306	1 047.9	13.7
M4L3J10	457	7.0	457	7.7	0.0	M4L4J10	738	10.4	730	11.8	1.1
M4L3J30	3 742	31.0	3 502	92.4	6.9	M4L4J30	8 942	44.1	7 712	107.1	15.9
M4L3J50	12 850	66.5	10 614	1 189.4	21.1	M4L4J50	35 850	102.6	29 038	991.8	23.5
M5L3J10	457	7.0	457	8.4	0.0	M5L4J10	774	10.7	730	13.3	6.0
M5L3J30	3 942	29.7	3 502	155.3	12.6	M5L4J30	9 226	44.5	7 712	177.2	19.6
M5L3J50	12 450	65.5	10 614	1 367.8	17.3	M5L4J50	35 642	105.7	29 038	1 122.2	22.7

4 结 语

本文研究的 RHFS 调度问题,考虑了实际生产中的队列约束,以及不同处理类型的机器.在使用列生成算法求解问题时,采用自适应加速策略改进算法,并建立多重决策的动态规划算法求解子问题,在分支定界的过程中构造列池并设计变异规则.最后对各种规模的实例进行仿真实验,验证算法的可行性.实验结果表明,本文所设计的列生成算法对中等规模的问题是有效的.

参考文献:

[1] Zhou B, Gao Z, Chen J. Scheduling algorithm of dual-armed cluster tools with residency time and reentrant constraints [J]. *Journal of Central South University*, 2014, 21 (1): 160 – 166.

[2] 周炳海, 石潇铭. 带重入约束的双集束型晶圆制造设备调度算法[J]. 东北大学学报(自然科学版), 2013, 34 (9): 1305 – 1309.

(Zhou Bing-hai, Shi Xiao-ming. Scheduling algorithm for double-cluster of wafer fabrication system with reentrant constraints[J]. *Journal of Northeastern University (Natural Science)*, 2013, 34 (9): 1305 – 1309.)

[3] Graves S C, Meal H C, Stefek D, et al. Scheduling of re-entrant flow shops[J]. *Journal of Operations Management*, 1983, 3 (4): 197 – 207.

[4] Hekmatfar M, Ghomi S M T F, Karimi B. Two stage reentrant hybrid flow shop with setup times and the criterion of minimizing makespan[J]. *Applied Soft Computing*, 2011, 11 (8): 4530 – 4539.

[5] Dugardin F, Yalaoui F, Amodeo L. New multi-objective

method to solve reentrant hybrid flow shop scheduling problem [J]. *European Journal of Operational Research*, 2010, 203 (1): 22 – 31.

[6] Cho H M, Bae S J, Kim J, et al. Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm [J]. *Computers & Industrial Engineering*, 2011, 61 (3): 529 – 541.

[7] Jeong B, Kim Y D. Minimizing total tardiness in a two-machine re-entrant flowshop with sequence-dependent setup times[J]. *Computers & Operations Research*, 2014, 47 (47): 72 – 80.

[8] 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]. *International Journal of Advanced Manufacturing Technology*, 2009, 42 (9/10): 963 – 973.

[9] Wu K, Zhao N, Gao L, et al. Production control policy for tandem workstations with constant service times and queue time constraints [J]. *International Journal of Production Research*, 2016, 54 (21): 6302 – 6316.

[10] Wu K. Classification of queueing models for a workstation with interruptions: a review [J]. *International Journal of Production Research*, 2014, 52 (3): 902 – 917.

[11] An Y J, Kim Y D, Choi S W. Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times [J]. *Computers & Operations Research*, 2016, 71 (1): 127 – 136.

[12] Nishi T, Izuno T. Column generation heuristics for ship routing and scheduling problems in crude oil transportation with split deliveries[J]. *Computers & Chemical Engineering*, 2014, 60 (2): 329 – 338.

[13] Tas D, Gendreau M, Dellaert N, et al. Vehicle routing with soft time windows and stochastic travel times: a column generation and branch-and-price solution approach [J]. *European Journal of Operational Research*, 2014, 236 (3): 789 – 799.



(上接第 1265 页)

[7] Baker A R, Greenaway A M, Ingram C W. A microwave digestion-based determination of low molecular weight organic acids in Bayer process liquor [J]. *Talanta*, 1995, 42 (10): 1355 – 1360.

[8] Power G, Loh J S C, Vernon C. Organic compounds in the processing of lateritic bauxites to alumina; part 2: effects of organics in the Bayer process [J]. *Hydrometallurgy*, 2012, 127/128: 125 – 149.

[9] Pan X L, Yu H Y, Tu G F, et al. Effects of precipitation

activity of desilication products (DSPs) on stability of sodium aluminate solution [J]. *Hydrometallurgy*, 2016, 165: 261 – 269.

[10] 潘晓林, 蒋涛, 侯宪林, 等. 拜耳法过程水合铝硅酸钠析出活性研究 [J]. 中国有色金属学报, 2017, 27 (8): 1748 – 1755.

(Pan Xiao-lin, Jiang Tao, Hou Xian-lin, et al. Precipitation activity of sodium aluminosilicate hydrate during the Bayer process [J]. *The Chinese Journal of Nonferrous Metals*, 2017, 27 (8): 1748 – 1755.)