DOI: 10.7641/CTA.2017.70465

带运输考虑的多阶段动态可重入混合流水车间调度

轩 华†, 李 冰, 王薛苑, 徐春秋

(郑州大学 管理工程学院, 河南 郑州 450001)

摘要: 可重入混合流水车间调度允许一个工件多次进入某些加工阶段, 它广泛出现在许多工业制造过程中, 如半导体制造、印刷电路板制造等. 本文研究了带运输时间的多阶段动态可重入混合流水车间问题, 目标是最小化总加权完成时间. 针对该问题, 建立了整数规划模型, 进而基于工件解耦方式提出了两种改进的拉格朗日松弛(LR)算法. 在这些算法中, 设计了动态规划的改进策略以加速工件级子问题的求解, 提出了异步次梯度法以得到有效的乘子更新方向. 测试结果说明了所提出的两种改进算法在解的质量和运行时间方面均优于常规LR算法, 两种算法都能在可接受的计算时间内得到较好的近优解.

关键词: 动态可重入混合流水车间; 运输时间; 拉格朗日松弛; 改进动态规划; 异步次梯度优化

引用格式: 轩华, 李冰, 王薛苑, 等. 带运输考虑的多阶段动态可重入混合流水车间调度. 控制理论与应用, 2018, 35(3): 357 – 366

中图分类号: TB49 **文献标识码**: A

Multi-stage dynamic reentrant hybrid flowshop scheduling with transportation consideration

XUAN Hua†, LI Bing, WANG Xue-yuan, XU Chun-qiu

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

Abstract: Reentrant hybrid flowshop scheduling is widely found in many industries such as semiconductor manufacturing and printed circuit board fabrication, where a job visits some processing stages for several times. A multi-stage dynamic reentrant hybrid flowshop problem with transportation time is studied with the objective of minimizing total weighted completion time. Then an integer programming model is formulated and two improved Lagrangian relaxation (LR) algorithms are presented based on job decoupling. In these algorithms, dynamic programming is improved to speedup the resolution of job-level subproblems and interleaved subgradient optimization is designed to obtain an effective multiplier updating direction. Testing results demonstrate that the two proposed LR algorithms outperform the traditional LR in terms of solution quality and running time. Both of the two algorithms could get better near-optimal schedules within an acceptable computational time.

Key words: dynamic reentrant hybrid flowshop; transportation time; Lagrangian relaxation; improved dynamic programming; interleaved subgradient optimization

Citation: XUAN Hua, LI Bing, WANG Xueyuan, et al. Multi-stage dynamic reentrant hybrid flowshop scheduling with transportation consideration. *Control Theory & Applications*, 2018, 35(3): 357 – 366

1 引言(Introduction)

在可重入混合流水车间, 有多个加工阶段, 至少有一个加工阶段有多台并行机, 每个工件依次经过这些阶段进行加工, 由于特殊制造需求或质量要求, 一些工件需重复进入某些阶段多次[1]. 可重入混合流水车间问题(reentrant hybrid flowshop problem, RHFSP)比一般的HFSP更为复杂, 已被证明是NP难题[2], 而且它在工业领域具有广泛的应用, 因此有必要提出有效的方法解决该问题.

近年来, 已有不少专家研究RHFSP. 为最小化ma-

收稿日期: 2017−07−08; 录用日期: 2017−11−23.

†通信作者. E-mail: hxuan@zzu.edu.cn; Tel.: +86 13523082320.

†本文责任编委: 薛安克.

教育部人文社会科学研究项目(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), 河南省高等学校重点科研项目(17A520058)资助.

Supported by the Humanities & Social Sciences Research Foundation of Ministry of Education of China (15YJC630148), the National Science Foundations of China (U1604150), the Distinguished Young Teacher Development Foundation of Zhengzhou University (1421326092) and the Key Project of Higher University in Henan Province (17A520058).

kespan, 对于两阶段RHFSP, 文献[3]提出了分支定界算法和启发式算法; 文献[4]针对带阻塞约束的情况提出了基于模糊逻辑控制器的自适应调整的混合遗传算法和带Cauchy分布的混合粒子群优化法. 对于多阶段RHFSP, 文献[5]考虑了第一阶段(包含多个工作站)不同层之间以及两阶段的工件之间的调整时间, 提出了一些启发式和超启发式算法以及一个混合遗传算法; 文献[6]考虑了时间窗约束, 结合模糊逻辑控制方法, 开发了一种自适应混合遗传算法.

为最小化总加权完成时间, 文献[7]提出了基于异步次梯度法的拉格朗日松弛算法进行求解; 文献[8]则考虑了晶圆生产的多次检查和修复, 设计了结合动态规划和次梯度法的拉格朗日松弛算法得到中小规模问题的近优解.

为最小化 makespan 和总拖期, 文献[9–10]分别提出了迭代Pareto贪婪算法和改进的基于学习的优化算法; 文献[11]则提出了模糊Lorenz蚁群系统求解其置换调度. 为最大化瓶颈利用率和最小化最大完成时间, 文献[12]利用Lorenz统治关系提出了新的多目标遗传算法. 针对动态RHFSP, 以系统输出、平均流程时间、平均拖期和总拖期工件数为目标函数, 文献[13]提出了一种基于决策树的实时调度机制.

如上所述, 关于RHFSP的研究多关注makespan问题, 而由于总加权完成时间目标有助于改善不同加工阶段间的物流平衡和加强它们之间的时间衔接等, 故近年来关于该目标的研究也越来越受重视, 既有研究(如文献[7–8])仅考虑HFSP的可重入特性, 而未考虑实际生产环境中的工件释放时间和运输时间, 这些特征的引入使得RHFSP更贴近于实际生产, 但也使其求解更为复杂. 因此, 考虑相邻加工阶段间的运输时间, 假定所有工件动态到达系统的初端, 提出了本文所研究的带运输考虑的动态可重入混合流水车间调度(multi-stage dynamic reentrant hybrid flow-shop scheduling with transportation consideration, MDRHFS-TC), 旨在找到最小化总加权完成时间的调度时间表.

就RHFSP的求解算法而言, 多为智能优化算法, 对于拉格朗日松弛(Lagrangian relaxation, LR)这类基于最优化的近似算法的研究成果颇少, 如上述文献[7–8]均采用了LR进行求解, 但前者未给出具体的子问题求解方法; 后者则使用动态规划得到子问题最优解, 但当测试到最大工件数50时所需运行时间较长. 由于LR具有较好的分解特性且能为解的质量提供评价标准(即对偶间隙), 因此, 本文设计改进LR算法以有效求解MDRHFS–TC.

2 问题建模(Problem formulation)

2.1 问题描述(Problem description)

在MDRHFS–TC中, 有n个待加工工件$\{J_1, J_2, \cdots, J_n\}$, 总加工阶段数为$S$. 每个工件必须按照$St_1, St_2, \cdots, St_S$的顺序依次加工, 每个加工阶段$f$都有$m_f$台同构并行机, 每个工件访问这些加工阶段$L$次, 如图1所示. 因此, 每个工件被加工$L \cdot S$次. 每台机器一次至多加工一个工件, 一个工件在任一时间段至多在一台机器上加工. 假定工件动态到达系统的第一个加工阶段, 即所有工件在第1层第1阶段的最早开工时间必须在其释放时间之后; 考虑到生产过程中前后工序之间常借助传送带、吊车等运输工具实现物料递转, 因此将运输时间与加工时间分开考虑, 以兼顾机器调度和工件运输的协调问题. 调度的目标是最小化总加权完成时间.

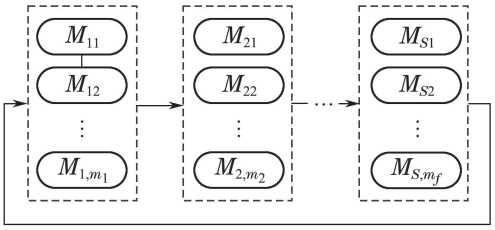


图 1 RHFS示意图

Fig. 1 The diagram of RHFS

2.2 数学模型(Mathematical model)

1) 参数.

n: 工件数;

S: 总加工阶段数;

L: 可重入层数;

m_f: 在阶段f上可利用的机器数;

P_{ilf}: 工件i在第l层的阶段f的加工时间;

$T_{(l,f),(l,f+1)}$: 同一层l内从阶段f到阶段$f+1$的运输时间;

$T_{(l,S),(l+1,1)}$: 从第l层的阶段S到第$l+1$层的阶段1的运输时间;

r_i: 工件i在第1层的第1阶段的释放时间;

K: 总计划时间范围;

α_i: 工件i的权重.

2) 决策变量.

c_{ilf}: 工件i在第l层的阶段f的完成时间.

X_{ilft}: 0/1变量, 当工件i在第l层的阶段f正在时刻t加工时其值为1, 否则为0.

利用上述符号, 可将MDRHFS–TC建模如下:

$$\min \sum_{i=1}^{n} \alpha_i c_{iLS}, \tag{1}$$

满足

$$\sum_{i=1}^{n}\sum_{l=1}^{L} X_{ilft} \leqslant m_f,\ f = 1, \cdots, S,\ t = 1, \cdots, K, \tag{2}$$

$$c_{i,l,f+1} - c_{i,l,f} - P_{i,l,f+1} \geqslant T_{(l,f),(l,f+1)},$$

$$i=1,\cdots,n,\ l=1,\cdots,L,\ f=1,\cdots,S-1, \quad (3)$$

$$c_{i,l+1,1}-c_{i,l,S}-P_{i,l+1,1}\geqslant T_{(l,S),(l+1,1)},$$
$$i=1,\cdots,n,\ l=1,\cdots,L-1, \quad (4)$$

$$c_{i11}-P_{i11}+1\geqslant r_i,\ i=1,\cdots,n, \quad (5)$$

$$X_{ilft}\in\{0,1\},\ i=1,\cdots,n,\ l=1,\cdots,L,$$
$$f=1,\cdots,S,\ t=1,\cdots,K. \quad (6)$$

$$c_{ilf}\in\{1,\cdots,K\},\ i=1,\cdots,n,\ l=1,\cdots,L,$$
$$f=1,\cdots,S. \quad (7)$$

式(1)表示最小化总加权完成时间; 式(2)确保每个时刻t正在第f个阶段加工的各层的工件数不会超过该时刻内该阶段上可利用的机器数; 式(3)–(4)表示同一层的同一工件的相邻两工序之间以及不同层之间相邻两工序之间的加工优先级关系, 即只有当前一工序完成加工并运送至下一阶段后, 后一工序才能开始加工; 式(5)说明了每个工件的释放时间约束; 式(6)–(7)为决策变量的取值范围. 该模型与以往文献[7–8]的不同之处在于引入了工件释放时间约束, 而且由于兼顾相邻加工阶段间的运输问题, 故需要在加工优先级约束中考虑同层或异层相邻两工序之间的运输时间.

3　基于拉格朗日松弛的优化求解策略(Optimization strategy based on Lagrangian relaxation)

松弛机器能力约束是求解混合流水车间问题的拉格朗日松弛算法中常用的一种方式, 即通过该约束的松弛从而将松弛问题分解为多个工件级子问题, 如文献[7–8]. 但是, 不同的子问题求解策略和乘子更新方法产生的解的质量和运算时间各有不同. 本节将从这两个方面对LR算法进行改进研究.

3.1　拉格朗日松弛改进(Improvement of Lagrangian relaxation)

引入非负拉格朗日乘子$\{u_{ft}\}$松弛约束(2), 可得到拉格朗日松弛问题(LR)

$$G(u)=$$
$$\min_{\{c_{ilf}\}}\{\sum_{i=1}^{n}\alpha_i c_{iLS}+\sum_{f=1}^{S}\sum_{t=1}^{K}u_{ft}(\sum_{i=1}^{n}\sum_{l=1}^{L}X_{ilft}-m_f)\}=$$
$$\min_{\{c_{ilf}\}}\{\sum_{i=1}^{n}(\alpha_i c_{iLS}+\sum_{l=1}^{L}\sum_{f=1}^{S}\sum_{t=1}^{K}u_{ft}X_{ilft})-$$
$$\sum_{f=1}^{S}\sum_{t=1}^{K}u_{ft}m_f\}, \quad (8)$$

满足式(3)–(7)和

$$u_{ft}\geqslant 0,\ f=1,\cdots,S,\ t=1,\cdots,K.$$

因此, 拉格朗日对偶问题(LD)为

$$\max G(u), \quad (9)$$

满足约束(3)–(8).

给定一组$\{u_{ft}\}$, 则LR问题可分解为n个工件级子问题:

$$G_i(c_{ilf})=\min_{\{c_{ilf}\}}\{\alpha_i c_{iLS}+\sum_{l=1}^{L}\sum_{f=1}^{S}\sum_{t=c_{ilf}-P_{ilf}+1}^{c_{ilf}}u_{ft}\}, \quad (10)$$

满足约束(3)–(8).

3.2　基于动态规划的工件级子问题求解(Solving job level subproblems using dynamic programming)

从第L层最后一个加工阶段S向第1层的第1个加工阶段依次计算每个工件的累计费用:

$$H(l,f,c_{ilf})=\alpha_i c_{iLS}+\sum_{t=1}^{K}u_{ft}X_{ilft},\ f=S,\ l=L, \quad (11)$$

$$H(l,f,c_{ilf})=\sum_{t=1}^{K}u_{ft}X_{ilft}+\min_{\{c_{i,l,f+1}\}}\{H(l,f+1,$$
$$c_{i,l,f+1})\},\ f=1,\cdots,S-1,\ l=1,\cdots,L, \quad (12)$$

$$H(l,f,c_{ilf})=\sum_{t=1}^{K}u_{ft}X_{ilSt}+\min_{\{c_{i,l+1,1}\}}\{H(l+1,1,$$
$$c_{i,l+1,1})\},\ f=S,\ l=1,\cdots,L-1. \quad (13)$$

3.3　基于改进动态规划的工件级子问题求解(Solving job level subproblems using improved dynamic programming)

对于第l'层第f'阶段(记为(l', f'))上相邻的两个节点z和z', 如果$z=z'+1$, 则它们在同一层的下一加工阶段$f'=f+1(f<S$时)或在下一层的第1阶段($f=S$时)对应的最早完成时间$z_$next 和$z'_$next也有同样的关系.

如图2所示, 令

$$A(z)=$$
$$\min_{\{c_{il'f'}\}}\{H(l',f',c_{il'f'})\}(c_{il'f'}\in\{z\_\text{next},\cdots,c_{l'f'}^{L'}\}),$$
$$A(z')=$$
$$\min_{\{c_{il'f'}\}}\{H(l',f',c_{il'f'})\}(c_{il'f'}\in\{z'\_\text{next},\cdots,c_{l'f'}^{L'}\}),$$

其中$c_{l'f'}^{L'}$为工件在第l'层的阶段f'上的最晚完成时间. 推广文献[14], 通过计算这两个节点之间的费用关系, 可将式(12)进一步表示为

$$H(l,f,c_{ilf})=$$
$$\min\{(u_{f,c_{ilf}-P_{ilf}+1}-u_{f,c_{ilf}+1})+$$
$$H(l,f,c_{ilf}+1),\sum_{t=c_{ilf}-P_{ilf}+1}^{c_{ilf}}u_{ft}X_{ilft}+$$

$$H(l,f+1,c_{ilf}+P_{i,l,f+1}+T_{(l,f),(l,f+1)})\},$$
$$c_{ilf}<c_{lf}^{L'},\ f<S,\ l\leqslant L, \quad (14)$$
$$H(l,f,c_{ilf})=$$
$$\sum_{t=c_{ilf}-P_{ilf}+1}^{c_{ilf}}u_{ft}X_{ilft}+H(l,f+1,c_{ilf}+$$
$$P_{i,l,f+1}+T_{(l,f),(l,f+1)}),\ c_{ilf}=c_{lf}^{L'},\ f<S,\ l\leqslant L. \quad (15)$$

同理, 式(13)可表示为

$$H(l,f,c_{ilf})=$$
$$\min\{(u_{f,c_{ilS}-P_{ilS}+1}-u_{f,c_{ilS}+1})+$$
$$H(l,f,c_{ilf}+1),\sum_{t=c_{ilS}-P_{ilS}+1}^{c_{ilS}}u_{ft}X_{ilft}+$$
$$H(l+1,1,c_{ilf}+P_{i,l+1,1}+T_{(l,S),(l+1,1)})\},$$
$$c_{ilf}<c_{lf}^{L'},\ f=S,\ l<L, \quad (16)$$
$$H(l,f,c_{ilf})=$$
$$\sum_{t=c_{ilS}-P_{ilS}+1}^{c_{ilS}}u_{ft}X_{ilft}+H(l+1,1,$$
$$c_{ilf}+P_{i,l+1,1}+T_{(l,f),(l+1,1)}),$$
$$c_{ilf}=c_{lf}^{L'},\ f=S,\ l<L, \quad (17)$$

其中

$$c_{lf}^{L'}=K-\sum_{h=f+1}^{S}P_{ilh}-\sum_{j=l+1}^{L}\sum_{h=1}^{S}P_{ijh}-$$
$$\sum_{h=f}^{S-1}T_{(l,h),(l,h+1)}-\sum_{j=l}^{L-1}T_{(j,S),(j+1,1)}.$$

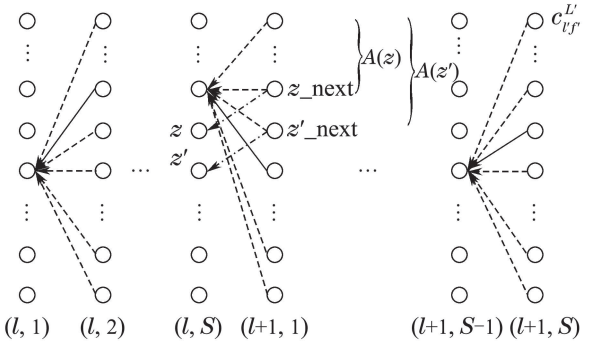


图 2 后向动态规划

Fig. 2 Backward dynamic programming

3.4 可行解的构造(Construction of feasible solutions)

由于求解松弛问题时松弛了机器能力约束, 因此对偶问题的解一般是原问题的不可行解. 因此, 基于松弛问题的解, 按照$(c_{i11}\sum_{l}\sum_{f}P_{ilf})/\alpha_i$ 的升序产生工件列表, 进而按此列表依次将各工件分配到第1层第一阶段可利用的机器上, 然后更新工件在下一阶段的可能开始时间, 按照其时间升序依次将各工件分配到可利用的机器上. 为进一步改进可行解, 当算法执行一定迭代数后, 基于两两交换策略改变工件顺序, 图3表示了由上述过程得到的当前可行时间表所对应的第l层第f阶段的工件序列, 从序列中排在第1个位置的工件开始, 将其与位于第$2,3,\cdots,n$的工件依次进行交换, 如果交换后得到的可行解优于当前所获得的最好解, 则更新上界和可行解; 否则恢复交换前的顺序. 同理, 再对位于第$2,\cdots,n-1$的工件依次执行上述交换过程, 直至完成每层每阶段的工件交换, 从而改善可行解, 记录得到的最佳可行解.

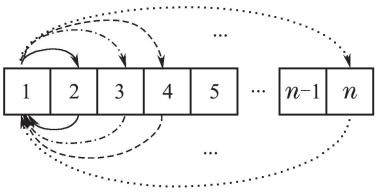


图 3 两两交换方法

Fig. 3 The pairwise interchange method

3.5 基于次梯度优化的对偶问题求解(Solving the dual problem using subgradient optimization)

对于拉格朗日乘子向量u, 求解对偶问题的次梯度算法为

$$u^{v+1}=u^v+\lambda^v g(u^v),\ v=0,1,2,\cdots, \quad (18)$$

其中: $g(u^v)$为$G(u^v)$的次梯度, $g(u^v)$等于$(\sum_i\sum_l X_{ilft}-m_f)$, λ^v为第v次迭代的步长, 给定如下:

$$\lambda^v=\beta\frac{G^*-G^v}{\|g(u^v)\|^2},\ 0<\beta<2. \quad (19)$$

3.6 基于异步次梯度优化的对偶问题求解(Solving the dual problem using interleaved subgradient optimization)

次梯度算法常用于求解拉格朗日对偶问题(如文献[8, 15–16]), 它要求每次迭代最优求解所有子问题, 因此当子问题求解较为耗时时, 其求解效率较低, 而且该算法的震荡现象引起迭代后期收敛速度较慢. 为了改进收敛性, 不少研究人员提出了改进策略, 如偏差条件ϵ次梯度级算法[17]、bundle法[18]、改进次梯度法[19]、异步次梯度法[14,20]等. 考虑实际生产的时间要求, 为减少每次迭代所需的时间以及降低最优求解所有子问题的复杂性, 本节提出异步次梯度优化方法, 基于异步迭代的思想, 在每次迭代仅最小化一个子问题, 而其他子问题的解保持为前一次迭代的值, 据此计算异步次梯度方向.

定义代理对偶为

$$\tilde{G}(u)=\sum_{i=1}^{n}\alpha_i c_{iLS}+\sum_{f=1}^{S}\sum_{t=1}^{K}u_{ft}(\sum_{i=1}^{n}\sum_{l=1}^{L}X_{ilft}-m_f).$$

异步次梯度法为

$$u^{v+1} = u^v + \theta^v \tilde{g}(u^v),\ v = 0, 1, 2, \cdots, \tag{20}$$

其中:

$$\tilde{g}(u^v) = \sum_i \sum_l X_{ilft} - m_f, \tag{21}$$

$$\theta^v = \beta \frac{G^* - \tilde{G}^v}{\|\tilde{g}(u^v)\|^2},\ 0 < \beta < 1. \tag{22}$$

3.7 3 种优化求解算法(Three optimization algorithms)

根据上述子问题的不同求解方法和乘子向量的不同更新方法, 设计3种求解算法:

1) DP&SO–LRJ1算法: 表示常规的基于一般动态规划(dynamic programming)和次梯度优化(subgradient optimization)的LR算法.

2) IDP&SO–LRJ2算法: 表示基于改进动态规划(improved dynamic programming)和次梯度优化的LR算法.

3) IDP&ISO–LRJ3算法: 表示基于改进动态规划和异步次梯度法(interleaved subgradient optimization)的LR算法, 其流程图如图4所示.

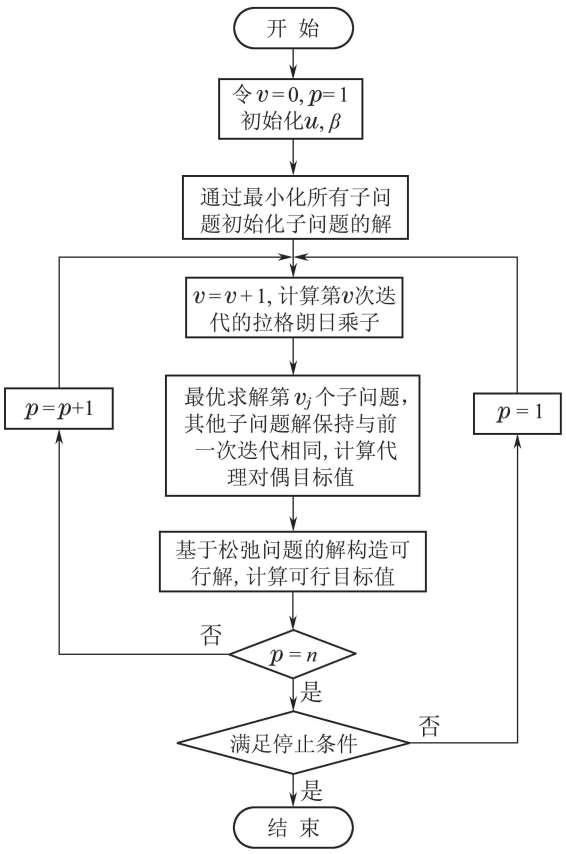


图 4 IDP&ISO–LRJ3算法流程图

Fig. 4 The flowchart of the IDP&ISO–LRJ3 algorithm

上述3种算法中, 分解后的工件级子问题数均为n个, 乘子数均为$S \times K$. 但DP&SO–LRJ1和IDP&SO–LRJ2每次迭代需求解的子问题数均为n个, 而IDP&ISO–LRJ3则为1个. 求解子问题时, DP&SO–LRJ1的动态规划中每个阶段的状态总数为K, 复杂度为$O(L(S-1)K^2)$; IDP&SO–LRJ2和IDP&ISO–LRJ3中, 由于改进动态规划利用了同一阶段相邻节点间的费用关系计算每个状态的费用, 因此, 计算的状态数有所减少, 其复杂度为$O(2L(S-1)K)$.

4 仿真实验(Simulation experiments)

4.1 实验数据的产生(Generation of experimental data)

为比较所提3种算法, 随机产生和测试多组数据. 上述3种算法所使用的参数如表1所示. 由$\{n, L, S, m_f\}$ 4种参数可产生144个不同的组合, 对于每个组合, 随机产生10个实例, 因此最后共测试了$8 \times 2 \times 3 \times 3 \times 10 = 1440$个不同的实例.

表 1 用于生成随机问题的参数

Table 1 Parameters for generating random problems

参数	数据产生
工件数	$\{10, 20, 30, 40, 50, 60, 80, 100\}$
阶段数	$\{3, 4\}$
重入层数	$\{2, 3, 4\}$
每阶段的机器数	$\{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, \cdots, 24\}$
加工时间	$U[10, 20]$
运输时间	$U[3, 6]$
释放时间	$U[1, 6]$
工件权重	$U[1, 10]$

4.2 算法的比较(Comparison of algorithms)

利用C语言对上述3种算法进行编程, 在Windows7操作系统(64 位)的计算机(Intel Core i5–5200 U 2.20 GHz) 上运行. 实验结果通过对偶间隙(Gap(%) = (UB−LB)/LB×100%, 其中: UB为得到的最佳上界值, LB为得到的最佳下界值)或代理对偶间隙(SGap(%) = (UB−LB′)/LB′×100%, 其中LB′为当前代理对偶目标值)、迭代数(ItN)和CPU时间(Time)来衡量. 为公平测试这3种算法, 使用相同的停止条件, 即当对偶间隙小于0.5%或迭代数达到500时, 程序停止. 考虑到实际生产环境中时间的重要性, 也将最大运行时间900 s作为一个停止条件.

表2列出了10个工件规模问题的3种算法的运行结果, 表中的每个值为$\{n, L, S, m_f\}$的10个实例的平均结果(除最后一行). 从表中可以看出, 3种算法都能得到高质量的解, 但相较于 IDP&SO–LRJ2 和 IDP&ISOLRJ3, DP&SO–LRJ1 所需的计算时间要长得多. DP&SO–LRJ1在平均676.47 s内得到0.93%的平均对偶间隙, 而 IDP&SO–LRJ2 和 IDP&ISO–LRJ3在平均4.14 s和3.02 s分别得到0.47%和0.95%的平均对偶间隙. 由于在DP&SO–LRJ1中每次迭代都要对每个工件

的所有可能开始加工时间计算其费用, 所以在有限的时间内运行的迭代数较少, 消耗的时间过长. 考虑到实际生产中时间的重要性, 故在测试更大规模问题时仅测试了IDP&SO–LRJ2和IDP&ISO–LRJ3.

表 2 $n = 10$时3种算法的计算结果

Table 2 Computational results of the three algorithms for $n = 10$

问题	n, L, S, m_f	DP&SO–LRJ1			IDP&SO–LRJ2			IDP&ISO–LRJ3			
		Gap	Time	ItN	Gap	Time	ItN	SGap	Gap	Time	ItN
1	10, 2, 3, 4	0.51	321.66	217.30	0.51	2.03	217.30	1.29	1.30	1.00	500.00
2	10, 2, 3, 5	0.42	274.04	183.50	0.42	1.61	183.50	1.18	1.17	0.97	500.00
3	10, 2, 3, 6	0.49	163.68	115.50	0.49	1.02	115.50	1.06	1.06	0.95	500.00
4	10, 3, 3, 4	0.64	731.41	138.50	0.40	4.33	223.10	0.94	0.94	2.10	500.00
5	10, 3, 3, 5	0.47	555.29	104.20	0.47	2.27	115.40	0.84	0.84	2.13	500.00
6	10, 3, 3, 6	0.49	559.97	106.00	0.49	2.08	106.00	0.83	0.83	2.01	494.80
7	10, 4, 3, 4	1.63	900.00	68.50	0.47	7.64	208.60	1.03	1.03	3.87	500.00
8	10, 4, 3, 5	1.25	900.00	67.90	0.47	3.49	97.50	0.71	0.71	3.52	500.00
9	10, 4, 3, 6	0.94	900.00	69.20	0.49	3.38	95.80	0.73	0.73	3.32	495.90
10	10, 2, 4, 4	0.50	583.64	156.90	0.49	2.79	179.30	1.31	1.31	1.68	500.00
11	10, 2, 4, 5	0.47	464.18	123.20	0.47	1.95	123.20	1.15	1.15	1.66	500.00
12	10, 2, 4, 6	0.49	422.64	113.70	0.49	1.77	113.70	1.07	1.07	1.65	500.00
13	10, 3, 4, 4	1.34	900.00	70.70	0.49	5.92	162.40	0.94	0.94	3.57	500.00
14	10, 3, 4, 5	1.18	900.00	69.90	0.49	3.72	105.20	0.84	0.84	3.35	492.90
15	10, 3, 4, 6	1.05	900.00	70.60	0.47	3.86	110.30	0.75	0.74	3.51	500.00
16	10, 4, 4, 4	2.38	900.00	29.30	0.47	12.88	200.40	0.95	0.95	6.36	500.00
17	10, 4, 4, 5	1.55	900.00	29.60	0.48	7.24	114.80	0.71	0.71	6.42	500.00
18	10, 4, 4, 6	1.02	900.00	29.40	0.48	6.54	103.30	0.69	0.69	6.37	500.00
平均		0.93	676.47	97.99	0.47	4.14	143.07	0.95	0.95	3.02	499.09

表3–4列出了多达100个工件规模问题的平均测试结果. 从表中可得到以下结论:

1) 对不同规模的问题, 两种算法均能在可接受的时间内得到较满意的近优解. 就平均性能而言, 由IDP&SO–LRJ2得到的解的质量优于IDP&ISO–LRJ3, 但IDP&ISO–LRJ3所需的运行时间比IDP&SO–LRJ2要短的多. IDP&SO–LRJ2 的对偶间隙比 IDP&ISO–LRJ3平均降低了1.74%, 而IDP&ISO–LRJ3的运行时间则比IDP&SO–LRJ2平均降低了105.79 s.

2) 对于20–50个工件规模的问题, IDP&SO–LRJ2在平均195.49 s内得到的总平均对偶间隙为1.04%; IDP&ISO–LRJ3在平均94.59 s内得到的总平均对偶间隙为2.51%, 代理对偶间隙为2.52%. 因此, 虽然IDP&SO–LRJ2的对偶间隙比IDP&ISO–LRJ3平均降低了1.47%, 但计算时间却比IDP&ISO–LRJ3多了一倍多.

3) 对于60–100个工件规模的问题, IDP&SO–LRJ2 在平均 727.65 s 内得到的总平均对偶间隙为3.14%; IDP&ISO–LRJ3 在平均615.34 s内得到的总平均对偶间隙为5.26%, 代理对偶间隙为5.20%. 因此, IDP&SO–LRJ2的对偶间隙比IDP&ISO–LRJ3平均降低了 2.12%, 但计算时间比 IDP&SO–LRJ2 多了112.31 s. 随着问题规模的增大, 算法所需运行时间也有所增加, 因此在测试大规模算例时, 常常算法还未运行到最大迭代数就因最大运行时间条件而终止, 故而, IDP&ISO–LRJ3与IDP&SO–LRJ2相比的时间优越性没有中小规模问题那么明显.

4) IDP&SO–LRJ2中, 当工件数、阶段数和机器数保持不变时, 对偶间隙随层次数的增加而变大. 这是因为更多的层次数会增加不同层次的工件对机器的竞争, 从而使问题更难于求解.

5) IDP&ISO–LRJ3中, 当阶段数较少时, 对偶间隙也随层次数增加而增大, 但当阶段数较多时, 随着层次数的增加, 对偶间隙反而呈下降趋势. 对不同规模的问题, 真正对偶间隙和代理对偶间隙之间的差距随机器数的增加而减少, 其最大、最小和平均差值分别为0.79, 0.00和0.03. 因此, 由IDP&ISO–LRJ3得到的代理对偶间隙与算法停止时计算得到的真正对偶间隙的差距很小.

以IDP&ISO–LRJ3所需的计算时间为停止条件运行IDP&SO–LRJ2, 图5列出了不同规模问题的$\{n, L, S, m_f\}$的5个实例:

$$(\{20,3,3,6\},\{40,3,3,10\},\{60,3,3,14\},\{80,3,3,18\},\{100,3,3,22\})$$

的(代理)对偶间隙变化趋势图. 从图中可看出, 在相同的计算时间内, 虽然随着问题规模的增大, IDP&ISO–LRJ3得到的代理对偶间隙略差于IDP&SO–LRJ2, 但IDP&ISO–LRJ3收敛速度较快, 所需的计算时间比IDP&SO–LRJ2短得多, 这在实际工业生产中尤为重要.

表 3 $n \in \{20, 30, 40, 50\}$ 时两种算法的平均对偶间隙和CPU时间

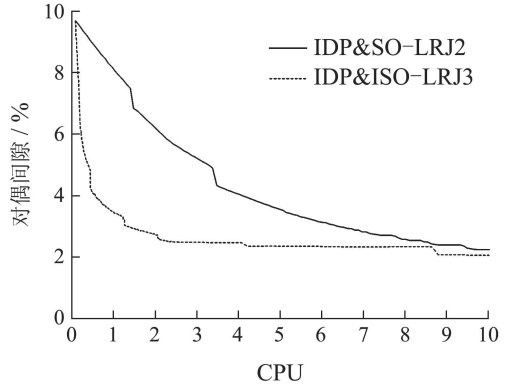
Table 3 Average duality gap and CPU time of two algorithms for $n \in \{20, 30, 40, 50\}$

问题	n, L, S, m_f	IDP&SO–LRJ2		IDP&ISO–LRJ3			问题	n, L, S, m_f	IDP&SO–LRJ2		IDP&ISO–LRJ3		
		Gap	Time	SGap	Gap	Time			Gap	Time	SGap	Gap	Time
1	20, 2, 3, 6	0.50	15.80	1.94	1.94	4.57	37	40, 2, 3, 10	2.74	106.88	5.17	5.19	29.87
2	20, 2, 3, 7	0.46	9.20	1.73	1.73	4.53	38	40, 2, 3, 11	0.73	87.30	2.77	2.77	29.46
3	20, 2, 3, 8	0.45	6.83	1.79	1.79	4.45	39	40, 2, 3, 12	0.46	52.39	2.46	2.46	29.44
4	20, 3, 3, 6	0.81	51.12	1.91	1.91	10.15	40	40, 3, 3, 10	3.80	238.92	5.56	5.60	72.88
5	20, 3, 3, 7	0.48	20.48	1.42	1.42	10.12	41	40, 3, 3, 11	1.54	239.59	3.24	3.24	74.16
6	20, 3, 3, 8	0.49	13.20	1.30	1.30	10.09	42	40, 3, 3, 12	0.49	163.56	2.02	2.02	75.66
7	20, 4, 3, 6	1.08	90.70	2.02	2.02	17.48	43	40, 4, 3, 10	5.03	442.14	6.52	6.62	130.90
8	20, 4, 3, 7	0.49	36.03	1.20	1.20	17.21	44	40, 4, 3, 11	2.08	433.64	3.45	3.46	134.56
9	20, 4, 3, 8	0.47	21.77	1.08	1.08	17.15	45	40, 4, 3, 12	0.49	353.27	1.91	1.91	136.19
10	20, 2, 4, 6	0.46	26.88	1.81	1.82	8.47	46	40, 2, 4, 10	0.44	125.50	2.47	2.47	59.92
11	20, 2, 4, 7	0.46	19.02	1.83	1.83	8.46	47	40, 2, 4, 11	0.46	103.63	2.36	2.36	58.85
12	20, 2, 4, 8	0.49	10.23	1.68	1.68	8.44	48	40, 2, 4, 12	0.46	70.48	2.27	2.27	58.98
13	20, 3, 4, 6	0.46	53.24	1.60	1.61	17.93	49	40, 3, 4, 10	0.48	297.17	2.03	2.03	141.95
14	20, 3, 4, 7	0.47	36.50	1.40	1.40	17.76	50	40, 3, 4, 11	0.48	175.22	2.00	2.00	135.67
15	20, 3, 4, 8	0.49	22.31	1.34	1.34	17.69	51	40, 3, 4, 12	0.48	181.83	1.76	1.76	136.56
16	20, 4, 4, 6	0.46	124.30	1.35	1.35	31.69	52	40, 4, 4, 10	0.46	574.84	1.82	1.82	228.70
17	20, 4, 4, 7	0.48	64.27	1.14	1.14	31.28	53	40, 4, 4, 11	0.44	386.72	1.67	1.67	226.98
18	20, 4, 4, 8	0.44	37.10	1.08	1.08	31.33	54	40, 4, 4, 12	0.49	201.36	1.48	1.48	224.36
19	30, 2, 3, 8	1.72	54.96	4.04	4.06	13.96	55	50, 2, 3, 12	3.96	170.23	6.50	6.53	64.66
20	30, 2, 3, 9	0.51	26.52	2.27	2.27	13.89	56	50, 2, 3, 13	1.36	171.05	3.79	3.79	64.25
21	30, 2, 3, 10	0.45	20.76	2.20	2.20	13.92	57	50, 2, 3, 14	0.72	125.11	2.94	2.94	63.43
22	30, 3, 3, 8	2.36	127.29	4.09	4.12	29.96	58	50, 3, 3, 12	5.37	385.69	7.29	7.36	136.55
23	30, 3, 3, 9	0.49	80.93	1.78	1.78	29.45	59	50, 3, 3, 13	2.35	381.32	4.27	4.27	135.71
24	30, 3, 3, 10	0.44	64.29	1.72	1.71	29.55	60	50, 3, 3, 14	0.73	364.49	2.66	2.66	129.39
25	30, 4, 3, 8	2.80	228.66	3.99	4.03	57.11	61	50, 4, 3, 12	6.61	702.33	8.21	8.42	226.55
26	30, 4, 3, 9	0.73	220.66	2.05	2.06	56.29	62	50, 4, 3, 13	2.96	728.54	4.47	4.49	243.82
27	30, 4, 3, 10	0.46	131.33	1.58	1.58	58.90	63	50, 4, 3, 14	1.14	728.85	2.75	2.76	243.94
28	30, 2, 4, 8	0.53	64.11	2.37	2.37	26.34	64	50, 2, 4, 12	0.47	217.72	2.75	2.75	112.83
29	30, 2, 4, 9	0.47	56.01	2.17	2.17	26.33	65	50, 2, 4, 13	0.45	158.27	2.56	2.56	112.02
30	30, 2, 4, 10	0.49	28.12	2.02	2.02	26.26	66	50, 2, 4, 14	0.46	126.19	2.60	2.60	112.15
31	30, 3, 4, 8	0.46	149.47	1.78	1.78	61.82	67	50, 3, 4, 12	0.49	553.52	2.29	2.29	265.32
32	30, 3, 4, 9	0.48	97.11	1.76	1.76	61.87	68	50, 3, 4, 13	0.46	368.98	2.14	2.14	249.13
33	30, 3, 4, 10	0.49	64.64	1.53	1.53	61.76	69	50, 3, 4, 14	0.47	232.97	2.03	2.03	245.29
34	30, 4, 4, 8	0.45	263.75	1.64	1.64	102.65	70	50, 4, 4, 12	0.53	818.20	1.91	1.91	425.00
35	30, 4, 4, 9	0.47	163.95	1.42	1.42	103.97	71	50, 4, 4, 13	0.46	608.71	1.74	1.74	422.16
36	30, 4, 4, 10	0.49	103.40	1.32	1.32	103.66	72	50, 4, 4, 14	0.49	393.78	1.67	1.67	426.43
								平均	1.04	195.49	2.51	2.52	94.59

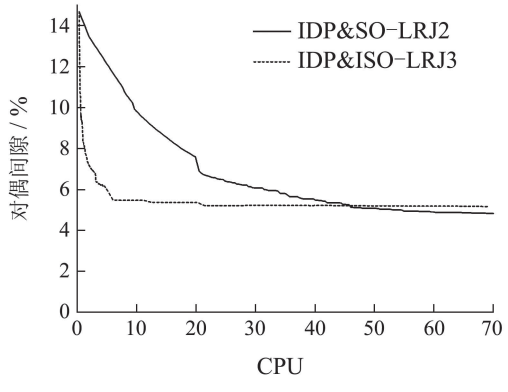
表 4 $n \in \{60, 80, 100\}$时两种算法的平均对偶间隙和CPU时间

Table 4 Average duality gap and CPU time of two algorithms for $n \in \{60, 80, 100\}$

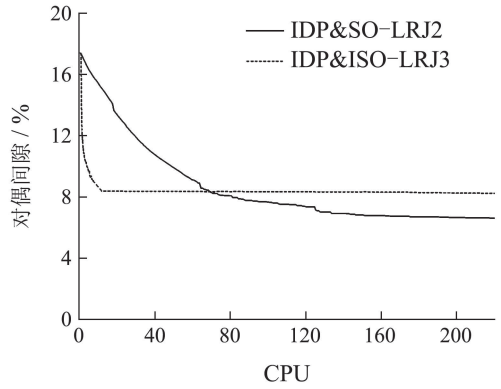
问题	n, L, S, m_f	IDP&SO–LRJ2		IDP&ISO–LRJ3			问题	n, L, S, m_f	IDP&SO–LRJ2		IDP&ISO–LRJ3		
		Gap	Time	SGap	Gap	Time			Gap	Time	SGap	Gap	Time
1	60, 2, 3, 14	4.82	277.35	7.52	7.55	109.87	28	80, 2, 4, 18	0.46	534.95	3.19	3.19	450.25
2	60, 2, 3, 15	2.32	278.76	4.90	4.91	109.55	29	80, 2, 4, 19	0.45	432.04	2.99	2.99	454.04
3	60, 2, 3, 16	1.12	278.26	3.66	3.66	108.50	30	80, 2, 4, 20	0.40	436.33	3.01	3.01	429.17
4	60, 3, 3, 14	6.68	632.84	8.86	9.00	235.55	31	80, 3, 4, 18	0.74	900.00	2.50	2.50	900.00
5	60, 3, 3, 15	3.52	630.65	5.69	5.70	243.81	32	80, 3, 4, 19	0.63	828.97	2.46	2.46	900.00
6	60, 3, 3, 16	1.54	640.55	3.59	3.59	243.56	33	80, 3, 4, 20	0.54	757.10	2.36	2.36	900.00
7	60, 4, 3, 14	7.98	900.00	9.79	10.04	440.10	34	80, 4, 4, 18	1.23	900.00	2.38	2.38	900.00
8	60, 4, 3, 15	4.22	900.00	6.04	6.06	428.43	35	80, 4, 4, 19	0.72	900.00	2.06	2.06	900.00
9	60, 4, 3, 16	1.99	900.00	3.56	3.56	421.97	36	80, 4, 4, 20	0.56	895.31	2.06	2.06	900.00
10	60, 2, 4, 14	0.48	318.27	2.84	2.84	204.39	37	100, 2, 3, 22	6.28	900.00	9.74	9.81	500.14
11	60, 2, 4, 7	0.45	215.61	2.92	2.92	201.06	38	100, 2, 3, 23	4.87	900.00	8.25	8.28	758.66
12	60, 2, 4, 16	0.49	182.05	2.75	2.75	216.04	39	100, 2, 3, 24	3.32	900.00	6.58	6.59	541.62
13	60, 3, 4, 14	0.47	672.04	2.43	2.43	425.58	40	100, 3, 3, 22	9.38	900.00	11.15	11.49	900.00
14	60, 3, 4, 15	0.48	539.26	2.16	2.16	440.82	41	100, 3, 3, 23	8.34	900.00	10.19	10.33	900.00
15	60, 3, 4, 16	0.46	444.56	2.07	2.07	429.00	42	100, 3, 3, 24	6.11	900.00	8.14	8.15	900.00
16	60, 4, 4, 14	0.75	879.91	2.01	2.01	738.67	43	100, 4, 3, 22	11.46	900.00	12.85	13.64	900.00
17	60, 4, 4, 15	0.55	767.42	1.88	1.88	748.23	44	100, 4, 3, 23	8.86	900.00	10.43	10.61	900.00
18	60, 4, 4, 16	0.50	659.37	1.78	1.78	728.12	45	100, 4, 3, 24	6.85	900.00	8.33	8.37	900.00
19	80, 2, 3, 18	5.63	528.60	9.09	9.15	245.26	46	100, 2, 4, 22	0.45	714.45	3.44	3.44	778.85
20	80, 2, 3, 19	4.13	553.07	7.39	7.41	264.77	47	100, 2, 4, 23	0.46	656.73	3.46	3.47	822.26
21	80, 2, 3, 20	2.34	550.98	5.35	5.35	262.33	48	100, 2, 4, 24	0.44	703.32	3.33	3.33	803.60
22	80, 3, 3, 18	8.74	900.00	11.14	11.41	549.21	49	100, 3, 4, 22	0.96	900.00	2.84	2.85	900.00
23	80, 3, 3, 19	4.97	900.00	7.64	7.67	539.57	50	100, 3, 4, 23	0.74	900.00	2.63	2.64	900.00
24	80, 3, 3, 20	3.27	900.00	5.68	5.69	531.08	51	100, 3, 4, 24	0.60	884.08	2.55	2.55	900.00
25	80, 4, 3, 18	10.19	900.00	11.50	11.95	876.21	52	100, 4, 4, 22	1.89	900.00	2.65	2.65	900.00
26	80, 4, 3, 19	6.71	900.00	8.00	8.08	890.15	53	100, 4, 4, 23	1.39	900.00	2.29	2.30	900.00
27	80, 4, 3, 20	5.20	900.00	6.45	6.46	857.94	54	100, 4, 4, 24	1.23	900.00	2.21	2.21	900.00
								平均	3.14	727.65	5.20	5.26	615.34



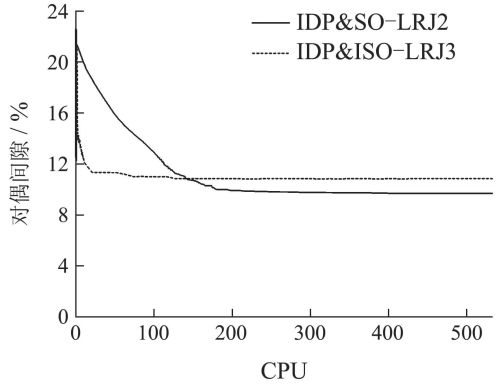
(a) 20个工件的对偶间隙变化趋势图



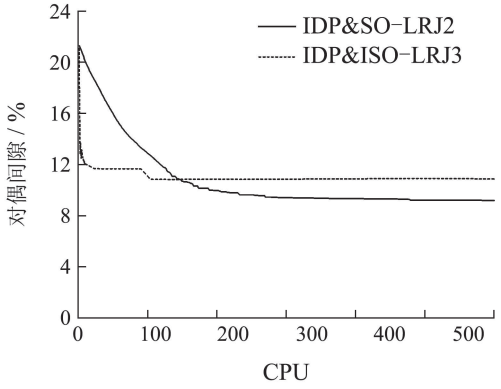
(b) 40个工件的对偶间隙变化趋势图



(c) 60个工件的对偶间隙变化趋势图



(d) 80个工件的对偶间隙变化趋势图



(e) 100个工件的对偶间隙变化趋势图

图 5 不同规模问题的对偶间隙变化趋势图

Fig. 5 Evolution of the duality gap of different sized problems

5 结论(Conclusions)

本文利用LR算法解决了带运输时间的多阶段动态可重入混合流水车间调度问题(MDRHFS–TC). 基于机器能力约束的松弛, 本文提出了两种改进策略, 即改进动态规划加速拉格朗日子问题的求解过程和设计异步次梯度法求解对偶问题, 从而形成两种改进算法, 即IDP&SO–LRJ2算法和IDP&ISO–LRJ3算法. 对大量随机生成的数据进行测试, 结果表明所提出的IDP&SO–LRJ2算法和IDP&ISO–LRJ3算法都优于传统的DP&SO–LRJ1算法, 而且都能在可接受的计算时间内得到高质量的解. 两种改进方法相比较而言, 结合改进动态规划和次梯度法的IDP&SO–LRJ2算法得到的解的质量最佳, 但运行时间较长; 而结合改进动态规划和异步次梯度法的IDP&ISO–LRJ3算法所需的运行时间较短, 但得到的近优解略差. 进一步的研究还可围绕所提方法的改进以及应用领域的扩展来展开.

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]. *International Journal of Production Research*, 2010, 49(8): 2221 – 2242.

[2] WANG M Y, SETHI S P, VAN DE VELDE S L. Minimizing makespan in a class of reentrant shops [J]. *Operations Research*, 1997, 45(5): 702 – 712.

[3] 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.

[4] SANGSAWANG C, SETHANAN K, FUJIMOTO T, et al. Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking constraint [J]. *Expert Systems with Applications*, 2015, 42(5): 2395 – 2410.

[5] HEKMATFAR M, FATEMI GHOMI S M T, 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.

[6] CHAMNANLOR C, SETHANAN K, CHIEN C F, et al. Re-entrant flow shop scheduling problem with time windows using hybrid genetic algorithm based on auto-tuning strategy [J]. *International Journal of Production Research*, 2014, 52(9): 2612 – 2629.

[7] JIANG S J, TANG L X. Lagrangian relaxation algorithms for re-entrant hybrid flowshop scheduling [C] //*International Conference on Information Management, Innovation Management and Industrial Engineering*. Taipei: IEEE, 2008: 78 – 81.

[8] ZHOU Binghai, ZHONG Zhenyi. Lagrangian relaxation algorithm for scheduling problems of reentrant hybrid flow shops [J]. *Control Theory & Applications*, 2015, 32(7): 881 – 886.
(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.)

[9] YING K C, LIN S W, WAN S Y. Bi-objective reentrant hybrid flowshop scheduling: An iterated Pareto greedy algorithm [J]. *International Journal of Production Research*, 2014, 52(19): 5735 – 5747.

[10] SHEN J N, WANG L, ZHENG H Y. A modified teaching-learning-based optimisation algorithm for bi-objective re-entrant hybrid flowshop scheduling [J]. *International Journal of Production Research*, 2016, 54(12): 3622 – 3639.

[11] FREDERIC D, LIONEL A, FAROUK Y. Fuzzy Lorenz ant colony system to solve multiobjective reentrant hybride flowshop scheduling problem [C] //*Interational Conference on Communications, Computing and Control Applications*. Tunisia: IEEE, 2011: 1 – 6.

[12] 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.

[13] CHOI H S, KIM J S, LEE D H. Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to TFT–LCD line [J]. *Expert Systems with Applications*, 2011, 38(4): 3514 – 3521.

[14] TANG L, XUAN H. Lagrangian relaxation algorithms for real-time hybrid flowshop scheduling with finite intermediate buffers [J]. *Journal of the Operational Research Society*, 2006, 57(3): 316 – 324.

[15] ZHOU Binghai, HU Liman. Lagrangian relaxation algorithm for material handling problems of assembly lines [J]. *Control Theory & Applications*, 2017, 34(4): 491 – 498.
(周炳海, 胡理嫚. 装配线物料搬运的拉格朗日松弛算法 [J]. 控制理论与应用, 2017, 34(4): 491 – 498.)

[16] YIN M, ZHOU B H, LU Z Q. An improved Lagrangian relaxation heuristic for the scheduling problem of operating theatres [J]. *Computers & Industrial Engineering*, 2016, 101: 490 – 503.

[17] PANG X F, GAO L, PAN Q K, et al. A novel Lagrangian relaxation level approach for scheduling steelmaking-refining-continuous casting production [J]. *Journal of Central South University*, 2017, 24(2): 467 – 477.

[18] JENA S D, CORDEAU J F, GENDRON B. Lagrangian heuristics for large-scale dynamic facility location with generalized modular capacities [J]. *Informs Journal on Computing*, 2017, 29(3): 388 – 404.

[19] SUN L L, LUAN F J, YING Y, et al. Rescheduling optimization of steelmaking-continuous casting process based on the Lagrangian heuristic algorithm [J]. *Journal of Industrial and Management Optimization*, 2017, 13(3): 1431 – 1448.

[20] XUAN Hua, LI Bing. Lagrangian relaxation using interleaved subgradient method and its application for multi-stage HFSP [J]. *Operations Research and Management Science*, 2015, 24(6): 121 – 127.
(轩华, 李冰. 基于异步次梯度法的LR算法及其在多阶段HFSP的应用 [J]. 运筹与管理, 2015, 24(6): 121 – 127.)

作者简介:

轩 华 (1979–), 女, 博士, 教授, 主要从事生产计划与调度、物流优化与控制等方面的研究, E-mail: hxuan@zzu.edu.cn;

李 冰 (1976–), 男, 博士, 教授, 主要从事物流优化与控制等方面的研究, E-mail: lbing@zzu.edu.cn;

王薛苑 (1985–), 男, 博士, 讲师, 主要从事物流优化与控制等方面的研究, E-mail: wangxueyuan@zzu.edu.cn;

徐春秋 (1983–), 女, 博士, 讲师, 主要从事物流优化与控制等方面的研究, E-mail: xuchunqiu@zzu.edu.cn.