文章编号: 1001-0920 (2013) 06-0829-08

并行机实时调度问题的LR & CG算法

汪恭书, 唐立新

(东北大学 信息科学与工程学院，沈阳 110819)

摘 要: 研究了目标函数为最小化总加权完成时间的并行机实时调度问题. 建立该问题混合整数规划模型, 并提出融合拉格朗日松弛 (LR) 和列生成 (CG) 的 LR & CG 混合算法. 该算法包含双重迭代, 在内环以次梯度法作为下界求解器和列生成器, 在外环通过求解限制主问题来获得影子价格以调节拉格朗日乘子. 计算实验结果表明, 在相同的计算时间内, LR & CG 能够比常规的 LR 算法获得更好的上界和下界, 表明了前者具有更好的收敛性能.

关键词: 并行机调度；拉格朗日松弛；次梯度；列生成；状态空间松弛

中图分类号: TP273 **文献标志码:** A

LR & CG algorithm for parallel machine real-time scheduling problem

WANG Gong-shu, *TANG Li-xin*

(College of Information Science and Engineering，Northeastern University，Shenyang 110819，China．Correspondent: TANG Li-xin，E-mail：lixintang@mail.neu.edu.cn)

Abstract: The parallel machine real-time scheduling problem with objective of minimizing total weighted completed time is investigated. The problem is formulated as a mixed integer programming model, and a LR & CG hybrid algorithm which combines Lagrangian relaxation(LR) with column generation(CG) together is proposed to solve it. The LR & CG algorithm contains double loops. In the inner loop, the subgradient method is executed to calculate lower bound and generate columns, and in the outer loop the restricted master problem is solved to get shadow price which is used to adjust Lagrangian multiples. The results of computational experiment show that the LR & CG algorithm can obtain tighter lower bound and higher quality upper bound than the conventional LR algorithm within the same computational time, which implies that the previous one has better convergent performance.

Key words: parallel machine scheduling；Lagrangian relaxation；subgradient；column generation；state space relaxation

0 引 言

并行机实时调度问题考虑将 n 个工件 $N=\{1,2,\cdots,n\}$ 分配到 m 台机器 $M=\{1,2,\cdots,m\}$ 上加工. 每个工件 $i\in N$ 具有到达时间 r_i, 截止时间 d_i, 权重 w_i 和在机器 k 上所需的加工时间 p_{ik}. 每台机器同一时刻只能加工一个工件, 每个工件在一台机器上加工, 加工过程不可中断且必须在给定的时间窗 $[r_i,d_i]$ 内开始和完成, 目标为最小化总加权完成时间. 并行机实时调度问题广泛存在于生产制造领域. 例如, 在炼铁到炼钢工序中可以将每罐铁水看作一个工件, 转炉看成机器. 由于高炉生产的复杂性导致铁水的出铁时间具有随机性, 对于转炉冶炼而言, 每罐铁水均有一个达到时间. 因为铁水是高温液态的, 其停留、运输和等待都将引起温降, 当温降超过容忍程度时, 将损坏运铁鱼类车的寿命并严重影响最终钢的钢级. 所以, 每罐铁水必须在给定的时间点之前完成冶炼, 这个时间点称为截止时间. 合理的铁水调度要求将每罐铁水分配到恰当的转炉上冶炼, 从而提高铁水入炉温度, 提高炼铁、炼钢效率. 很多实际问题都可以归结为并行机实时调度问题, 因此对其进行深入研究具有重要实际意义. 在同构并行机的环境下, 所有机器具有相同的速度, 同一工件在不同机器上的加工时间是相同的, 即 $p_{ik}=p_i$; 在无关并行机环境下, 工件 i 在不同机器上的加工时间是任意的, 无特殊规律. 采用调度上通用的三域表示法, 同构并行机实时

收稿日期: 2012-06-12；**修回日期:** 2012-10-02.

基金项目: 国家自然科学基金重点项目(71032004)；国家自然科学基金青年基金项目(71202151)；高等学校博士学科点专项科研基金项目(20090042120038).

作者简介: 汪恭书(1980−), 男, 讲师, 博士, 从事流程工业生产与物流调度的研究；唐立新(1966−), 男, 教授, 博士生导师, 从事生产调度、物流管理等研究.

调度问题可以表示为 $Pm|r_i,d_i|\sum_{i\in N}w_iC_i$, 无关并行机实时调度问题可以表示为 $Rm|r_i,d_i|\sum_{i\in N}w_iC_i$. 本文主要研究 $Rm|r_i,d_i|\sum_{i\in N}w_iC_i$ 的最优化算法. 由于 $Pm|r_i,d_i|\sum_{i\in N}w_iC_i$ 是 $Rm|r_i,d_i|\sum_{i\in N}w_iC_i$ 的一种特殊情况, 显然针对 $Rm|r_i,d_i|\sum_{i\in N}w_iC_i$ 的最优化算法也可以用于求解 $Pm|r_i,d_i|\sum_{i\in N}w_iC_i$.

并行机实时调度问题的关键特征是工件必须在给定的时间窗口内开始和完成加工. 对于与时间窗相关的并行机调度问题, 学术界已有相关研究. Hoogeveen 等[1]将工件的共同交货期设定为一个时间窗, 该时间窗的大小与工件的加工时间成比例, 给出了一个动态规划算法. De 等[2]说明了将工件的共同交货期看作决策变量时, 如果机器数固定则有一个伪多项式时间的动态规划算法. Cheng 等[3]对带提前/拖期惩罚的并行机调度问题的一种特殊情况 (工件加工时间相同) 提出多项式时间算法. Arkin 等[4]考虑了工件具有固定开始和结束时间的并行机器调度问题, 并提出复杂度为 $O(n^2\log n)$ 精确算法. Bouzina 等[5]研究了与文献 [4] 类似的调度问题, 当目标函数为最大化被调度工件的个数时, 提出了复杂度为 $O(n\max\{\log n,m\})$ 的算法; 当目标函数为最大化被调度工件总权重时, 建立了考虑 $n+1$ 个节点和 $2n$ 个弧的最小费用流问题模型. 上述研究虽然都提出了多项式时间的精确算法, 但解决的都是一些特殊情况. 在实际中特殊情况发生的概率较小, 因此, 上述精确算法应用范围有限.

因为一种简单情况 $P2|r_i,d_i|\sum_{i\in N}w_iC_i$ 已被证明是 NP- 难问题[6], 显然 $Rm|r_i,d_i|\sum_{i\in N}w_iC_i$ 也是 NP- 难问题. NP- 难问题的研究主要集中于近似算法. Bar-Noy 等[7]针对工件带有到达时间的并行机调度问题提出了一个近似算法, 并给出了算法的最坏性分析. 文献 [8] 给出了两台同构并行机上在线批调度问题的近似算法, 并分析了算法的最坏情况. 文献 [9] 研究了运输阶段具有等待时间限制的成批运输和并行机生产协调调度问题, 提出了启发式算法并证明其最坏情况性能比. 这些算法能够在理论上证明近似比, 但近似比通常较大, 在实际中难以应用. 还有一类算法是基于仿生学的人工智能算法. 如 Sivrikaya 等[10]采用遗传算法解决调整时间与顺序相关的并行机调度问题. Driessel 等[11]将变邻域搜索算法应用于求解调整时间与顺序相关、具有优先级约束和工件到达时间的并行机调度问题. 智能优化算法虽然具有易实现和计算复杂度低等优点, 但解的质量无法评价. 能够在合理的时间内找到实际规模问题的最优解或可量化指标的近优解, 这类算法更受青睐. 列生成和拉格朗日松弛属于这一类算法, 它们通过松弛复杂约束将原问题分解为易于求解的子问题, 由此得到原问题的界, 然后基于下界信息构造可行解或执行分支搜索获得最优解, 因为能够获得较紧的界, 所以可以对解的质量进行评价. 由于这两种算法在求解组合优化问题时所表现出的卓越性能, 已广泛应用于求解并行机调度问题[12-17], 但没有考虑工件的时间窗, 并且也鲜有文献研究两种算法的混合策略.

本文针对并行机实时调度问题, 建立了混合整数规划模型, 并提出了基于拉格朗日松弛 (LR) 的分解策略. 由于常规的次梯度法存在收敛速度慢等缺陷, 本文将基于次梯度的 LR 算法与列生成 (CG) 算法相结合, 形成 LR & CG 混合算法, 以共同求解拉格朗日对偶问题, 从而提高收敛速度. 同时实现了常规基于次梯度的 LR 算法, 以测试 LR & CG 算法的性能. 实验结果表明, 在相同的计算时间内, LR & CG 算法的上界、下界和对偶间隙优于常规的 LR 算法.

1 混合整数规划模型

首先定义描述问题参数和变量符号.

$N=\{1,2,\cdots,n\}$ 为工件集, n 为工件总数; $M=\{1,2,\cdots,m\}$ 为机器集, m 为机器总数; r_i, d_i 和 w_i 分别为工件 i 的到达时间、截止时间和权重; p_{ik} 为工件 i 在机器 k 上所需的加工时间. 因为工件必须在给定的时间窗口内开始和完成加工, 对于任意满足条件 $d_i-p_{ik}>r_j+p_{jk}$ 的工件 i 和 j, 在机器 k 上 j 不能于 i 之前加工. 所以, 可以根据工件的时间窗口进行工件顺序约束的预处理. 对于任意工件 i 和机器 k, 确定一组在机器 k 上可以排在 i 之后的工件, 记为 $A_{ik}=\{j\in N|r_i+p_{ik}+p_{jk}\geqslant d_j\}$; 同理确定一组在机器 k 上可以排在 i 之前的工件, 记为 $B_{ik}=\{j\in N|d_i-p_{ik}-p_{jk}\leqslant r_j\}$. 为了便于模型表达, 记 $A'_{ik}=A_{ik}\bigcup 0$, $B'_{ik}=B_{ik}\bigcup 0$.

若工件 j 紧随着工件 i 在机器 k 上加工, 则 $x_{ijk}=1$; 否则 $x_{ijk}=0$, $\forall k\in M$, $i\in N$, $j\in A_{ik}$. 特别地, $x_{0jk}=1$表示工件 j 是机器 k 上的第一个工件, $x_{j0k}=1$ 表示工件 j 是机器 k 上的最后一个工件. C_{ik} 为工件 i 在机器 k 上的完成时间.

并行机实时调度问题的混合整数规划 (MIP) 模型描述如下:

$$z_{\mathrm{MIP}}=\min\sum_{i\in N}\sum_{k\in M}w_iC_{ik}. \tag{1}$$

$$\text{s.t.} \sum_{k\in M}\sum_{j\in A'_{ik}} x_{ijk}=1,\ \forall i\in N; \tag{2}$$

$$\sum_{j\in N} x_{0jk}\leqslant 1,\ \forall k\in M; \tag{3}$$

$$\sum_{j\in A'_{ik}} x_{ijk}=\sum_{j\in B'_{ik}} x_{jik},\ \forall k\in M,\ i\in N; \tag{4}$$

$$C_{jk}\geqslant p_{jk}+\mathrm{Big}_M(x_{0jk}-1),\ \forall k\in M,\ j\in N; \tag{5}$$

$$C_{jk}\geqslant C_{ik}+p_{jk}+\mathrm{Big}_M(x_{ijk}-1),\ \forall k\in M,\ j\in N,\ i\in B_{jk}; \tag{6}$$

$$C_{ik}\geqslant (r_i+p_{ik})\sum_{j\in A'_{ik}} x_{ijk},\ \forall k\in M, i\in N; \tag{7}$$

$$C_{ik}\geqslant d_i\sum_{j\in A'_{ik}} x_{ijk},\ \forall k\in M,\ i\in N; \tag{8}$$

$$x_{0jk},x_{j0k},x_{ijk}\in\{0,1\},\ \forall k\in M,\ j\in N,\ i\in B_{jk}. \tag{9}$$

目标函数 (1) 为最小化所有工件的总加权完成时间. 约束 (2) 保证每个工件只能被分配到一台机器上加工. 约束 (3) 表示每台机器的第 1 个位置最多允许安排一个工件. 约束 (4) 保证工件在机器上的安排是恰当的, 对应网络流优化问题中的流守恒约束. 约束 (5) 表示如果工件被安排到机器的第 1 个位置, 则其完成时间不小于加工时间, 其中 Big_M 表示一个足够大的正数. 约束 (6) 表示如果工件 j 紧随工件 i 在机器 k 上加工, 则工件 j 的完成时间不小于工件 i 的完成时间加上工件 j 的加工时间, 保证同一时刻在同一台机器上最多只能加工一个工件. 约束 (7) 和 (8) 表示当工件 i 分配到机器 k 上时, 工件 i 在机器 k 上的完成时间要满足时间窗口约束. 约束 (9) 定义了变量的取值范围.

2　拉格朗日松弛

在 MIP 模型中, 约束 (3) ∼ (8) 保证了每台机器上子调度的可行性, 仅约束 (2) 耦合了工件在不同机器上的分配, 因此 MIP 是可以分解的. 引入拉格朗日乘子向量 $\mu=(\mu_1,\mu_2,\cdots,\mu_n)\in\boldsymbol{R}^n$, 其中 μ_i 对应约束 (2) 中的第 i 行, 将约束 (2) 松弛到目标函数 (1) 中, 经过整理后可得到如下拉格朗日松弛 (LR) 问题满足约束 (3) ∼ (9):

$$z_D(\mu)=\min\sum_{k\in M}\sum_{i\in N}\Big(w_iC_{ik}-\mu_i\sum_{j\in A'_{ik}} x_{ijk}\Big)+\sum_{i\in N}\mu_i. \tag{10}$$

对于给定乘子 μ, LR 问题可以分解为 m 个独立子问题, 对应 m 台机器上的子调度. 定义机器 k 的子问题为 $z_k(\mu)$, 描述为

$$z_k(\mu)=\min\sum_{i\in N}\Big(w_iC_{ik}-\mu_i\sum_{j\in A'_{ik}} x_{ijk}\Big), \tag{11}$$

并满足约束 (3) ∼ (9). 单机子问题 $z_k(\mu)$ 的本质是从工件集 N 中找到一组工件子集, 并确定子集中的工件在机器 k 上的调度, 使得子集中的工件满足时间窗口约束, 并使式 (11) 达到最优. 采用三域表示法, 单机子问题可表示为 $1|r_i,d_i|\sum_{i\in N}(w_iC_i-\mu_i)a_i$, 其中 a_i 为决策工件 i 是否被调度的二元变量. 单机子问题仍然是 NP- 难问题, 因为两种简单情况 $1|r_i|\sum_{i\in N}w_iC_i$ 和 $1|d_i|\sum_{i\in N}w_iC_i$ 已被证明是 NP- 难问题.

令 $S\subseteq N$ 为任意工件子集, 状态 (S,t) 表示 S 中的工件在机器上仅被调度一次且最后一个工件的完成时间为 t. 函数 $f_k(S,t)$ 定义为机器 k 在状态 (S,t) 上的最小费用, 则单机子问题的动态规划递归方程为

$$f_k(S,t)=\begin{cases}\min\limits_{i\in S}\{g_k(S\setminus\{i\},t-p_{ik})+w_it-\pi_i\},\\ \qquad t\in[r_i+p_{ik},d_i];\\ \infty,\ \text{otherwise.}\end{cases}$$

$$g_k(S,t)=\min\{f_k(S,\tau|\tau\leqslant t)\}. \tag{12}$$

从递归方程可以看出, 动态规划算法的计算复杂度为 $O(n2^n)$, 是工件数的指数形式. 指数复杂度的动态规划算法不能有效求解单机子问题.

应用状态空间松弛技术将原状态空间松弛到新的状态空间. 在新状态空间中, 状态 (i,t) 表示工件 i 为最后被调度工件且完成时间为 t. 令函数 $f_k(i,t)$ 为机器 k 在状态 (i,t) 上的最小费用, 可以构造新的动态规划递归方程为

$$f_k(i,t)=\min\{g_k(j,t-p_{ik})+w_it-\mu_i|j\in B'_{ik}\},\ \forall i\in N,\ t\in[r_i+p_{ik},d_i];$$

$$g_k(j,t)=\min\{f_k(j,\tau)|\tau\leqslant t\}. \tag{13}$$

边界条件为

$$f_k(j,t)=\begin{cases}0,\ j=0,t=0;\\ \infty,\ j\in N,t<r_j+p_j\ \text{or}\ t>d_j.\end{cases} \tag{14}$$

可以证明, 新的状态空间状态数的界为 nT, $T=\max\{d_i-r_i-p_{ik}+1|i\in N\}$, 动态规划递归方程时间复杂度上限为 $O(n^2T^2)$, 是伪多项式时间算法. 函数 $f_k(i,t)$ 的下标 k 只是为了区分不同机器的单机子问题, 不代表阶段数, 阶段数由 i 表示. 需要指出的是, 原状态空间内的任何一个状态 (S,t) 都可以回溯到一个可行的单机子调度, 但是新状态空间内的状态 (i,t) 可能回溯到可行调度, 也可能回溯到不可行的伪调度. 令序列 $(i_1,i_2,\cdots,i_h)$ 为一个单机子调度, 如果对于任意 $a\neq b\in\{1,2,\cdots,h\}$, 均满足 $i_a\neq i_b$, 则该单机子调度为可行调度, 否则为伪调度. 新的状态空间除了包含可行调度的状态外还包含伪调度对应的状态,

因此新状态空间上的最优值为单机子问题提供一个下界, 即

$$f_k(n,D)\leqslant z_k(\mu),\tag{15}$$

其中 $D=\max\{d_i|i\in N\}$. 因此, 对于任意给定的给定乘子, 采用状态空间松弛求解 m 个单机子问题, 得到原问题的下界为

$$\mathrm{LB}(\mu)=\sum_{k\in M}f_k(n,D)+\sum_{i\in N}\mu_i\leqslant$$
$$\sum_{k\in M}z_k(\mu)+\sum_{i\in N}\mu_i=z_D(\mu)\leqslant z_{\mathrm{MIP}}.\tag{16}$$

拉格朗日对偶 (LD) 问题定义为

$$z_{\mathrm{LD}}=\min\{z_D(\mu)|\mu\in\boldsymbol{R}^n\}.\tag{17}$$

LD 问题的核心是寻找最优乘子使得 $z_D(\mu)$ 达到最大, 次梯度是常见的寻找最优乘子的近似方法, 采用次梯度法求解 LD 问题的步骤如下.

Step 1: 针对给定的乘子 μ, 采用动态规划 (13)～(14) 求解 m 个单机子问题, 得到下界 $\mathrm{LB}(\mu)$, 若 $\mathrm{LB}(\mu)$ 大于历史最好下界 LB^*, 则 $\mathrm{LB}^*=\mathrm{LB}(\mu)$.

Step 2: 基于 m 个单机子问题的解, 采用启发式构造原问题的可行解, 得到上界 UB; 如果 UB 小于历史最好上界 UB^*, 则 $\mathrm{UB}^*=\mathrm{UB}$.

Step 3: 如果对偶间隙 $(\mathrm{UB}^*-\mathrm{LB}^*)/\mathrm{LB}^*$ 小于给定阈值, 或达到最大迭代数, 则停止.

Step 4: 采用次梯度法更新乘子, 有

$$\mu_i=\mu_i+\lambda g_i(\mu)\frac{\mathrm{UB}^*-\mathrm{LB}(\mu)}{||g(\mu)||^2}.$$

其中: 次梯度 $g_i(\mu)=1-\sum_{k\in M}\sum_{j\in A'_{ik}}x_{ijk}$, 步长 λ 初始化为 0.5. 如果 UB^* 在连续迭代中均没有改进, 则将 λ 乘一个因子以调整搜索方向, 返回 Step 1.

3 LR & CG 算法

对于任意机器 k, 定义 Ω_k 为满足约束 (3)～(8)的所有单机子调度集. 机器 k 的单机子调度 $\omega\in\Omega_k$ 用变量 $x^\omega_{0jk},x^\omega_{j0k},x^\omega_{ijk},C^\omega_{jk}(k\in M,j\in N,i\in B_{jk})$ 描述. 定义 $a_{i\omega}$ 为工件 i 在单机子调度 ω 出现的次数, c_ω 为 ω 中所包含工件的加权完成时间之和, 则有

$$a_{i\omega}=\sum_{j\in A'_{ik}}x^\omega_{ijk},\ \forall i\in N,k\in M,\omega\in\Omega_k;\tag{18}$$

$$c_\omega=\sum_{j\in N}w_jC^\omega_{jk},\ \forall k\in M,\omega\in\Omega_k.\tag{19}$$

基于上述符号的定义, LD 问题可以采用以下线性规划模型描述:

$$z_{\mathrm{LD}}=\max\sum_{k\in M}\theta_k+\sum_{i\in N}\mu_i.\tag{20}$$

$$\text{s.t. }\theta_k\leqslant c_\omega-\sum_{i\in N}a_{i\omega}\mu_i,\ \forall k\in M,\omega\in\Omega_k;\tag{21}$$

$$\theta_k(\forall k\in M),\mu_i(\forall i\in N)\text{ Unrestricted.}\tag{22}$$

令 λ^k_ω 为约束 (21) 对应的对偶解, 则 LD 的线性对偶问题 (称为主问题 MP) 可以表示为

$$z_{\mathrm{MP}}=\min\sum_{k\in M}\sum_{\omega\in\Omega_k}c_\omega\lambda^k_\omega.\tag{23}$$

$$\text{s.t. }\sum_{k\in M}\sum_{\omega\in\Omega_k}a_{i\omega}\lambda^\omega_k=1,\ \forall i\in N;\tag{24}$$

$$\sum_{\omega\in\Omega_k}\lambda^\omega_k=1,\ \forall k\in M;\tag{25}$$

$$\lambda^\omega_k\geqslant 0,\ \forall k\in M,\omega\in\Omega_k.\tag{26}$$

MP 中的每一列代表一个可行的单机子调度. 当 λ^ω_k 为整数变量时, MP 与 MIP 等价. 实际上, 对 MIP 进行 Dantzig-Wolfe 分解后的线性松弛即为 MP. 对于每台机器 k 而言, 可行单机子调度集 Ω_k 的基数为问题规模的指数形式, 因此, MP 的列数也为问题规模指数形式. 求解 MP 时不可能枚举出所有的列, 对于这类模型而言, 列生成是一个较好的选择, 它从一个包含部分列变量的限制主问题 (RMP) 出发, 通过 RMP 和价格子问题 (SP) 间的迭代获得 MP 的最优解.

令 $\bar\mu_i$ 和 $\bar\theta_k$ 分别为 MP 中与约束 (24) 和 (25) 对应的对偶解 (影子价格), 为了寻找负消减费用列, 需要求解以下价格子问题 (SP):

$$\min\Big\{c_\omega-\sum_{i\in N}a_{i\omega}\bar\mu_i-\bar\theta_k|k\in M,\omega\in\Omega_k\Big\}.\tag{27}$$

SP 可按机器分解, 每个机器子问题记为 SP_k, 有

$$\min\Big\{c_\omega-\sum_{i\in N}a_{i\omega}\bar\mu_i-\bar\theta_k|\omega\in\Omega_k\Big\}.\tag{28}$$

将影子价格 $\bar\mu$ 当作拉格朗日乘子, 则除常数 $\bar\theta_k$ 外, SP_k 与单机子问题 $z_k(\mu)$ 等价. 因此, 基于状态空间松弛的动态规划算法也适用于求解价格子问题.

无论是基于次梯度的 LR 算法还是列生成算法均可以求解 LD 问题. 两者之间存在相似性也存在区别: 相同点在于两者均通过主问题和子问题之间的迭代来求解 LD 的最优解, 且两者的子问题结构完全相同; 区别在于乘子 (或影子价格) 更新方式不同, 前者采用次梯度法更新乘子, 后者通过求解 RMP 获得影子价格. 此外, 次梯度法是求解 LD 问题的一种近似算法, 主要优点在于计算简单且容易实现; 缺点在于迭代后期的振荡效应且不能保证求得 LD 的最优解. 列生成算法是求解 LD 线性对偶问题 MP 的一种精确算法, 优点在于当价格子问题不能生成负消减费用列时能保证求得 MP 问题最优解 (由线性规划强对偶性质可知此时也求得了 LD 的最优解); 缺点在于用单纯形求解 RMP 时计算复杂, 且存在尾效应, 导致算法迭代后期收敛较慢. 另一方面, 在次梯度法的迭代过程中, 松弛问题的解由于违背约束 (2), 对原问题

而言是不可行的, 需要采用启发式构造原问题的可行解. 在列生成算法迭代过程中, 如果 RMP 解是整数解, 则该解也是原问题的一个可行解, 即使 RMP 解是分数解, 由于不违背约束条件, 该分数解更容易构造可行解. 进一步, 如果在列生成算法迭代终止时, 得到 RMP 的解为整数, 则该解也是原问题的最优解. 最后需特别指出, LD 问题最优值 z_{LD} 与原问题最优值 z_{MIP} 之间的间隙属于固定间隙. 对于某些实例, 这个固定间隙可能较大, 因此导致次梯度法中上界和下界的间隙在迭代中也较大, 难以评价下界和上界的质量. 但线性规划对偶理论提供了另外一种对偶间隙的判定标准. 在次梯度法的每一步迭代中, 可得到 LD 问题的一个下界 $\mathrm{LB}(\mu)$; 在列生成算法中, RMP 仅包含部分列, 因此 z_{RMP} 为 MP 提供一个上界. LD 和 MP 互为线性对偶, 由线性规划弱对偶和强对偶性质可知 $\mathrm{LB}(\mu) \leqslant z_{\mathrm{RMP}}$, 当 $\mathrm{LB}(\mu)$ 和 z_{RMP} 分别为 LD 和 MP 的最优解时, $z_{\mathrm{LD}} = \mathrm{LB}(\mu) = z_{\mathrm{RMP}} = z_{\mathrm{MP}}$. 因此, 若 $\mathrm{LB}(\mu)$ 和 z_{RMP} 的间隙较小, 则可判定 $\mathrm{LB}(\mu)$ 已接近 LD 的最优解 z_{LD}.

综上所述, 次梯度法和列生成算法各有优缺点, 且它们在迭代过程中涉及相同的子问题, 由此受到启发, 可以在次梯度法迭代寻找下界 LB^* 时, 将单机子问题的解转换为 RMP 的列并加入 RMP. 基于上述分析, 提出次梯度和列生成的混合策略, 称为 LR & CG 算法, 流程如图 1 所示.

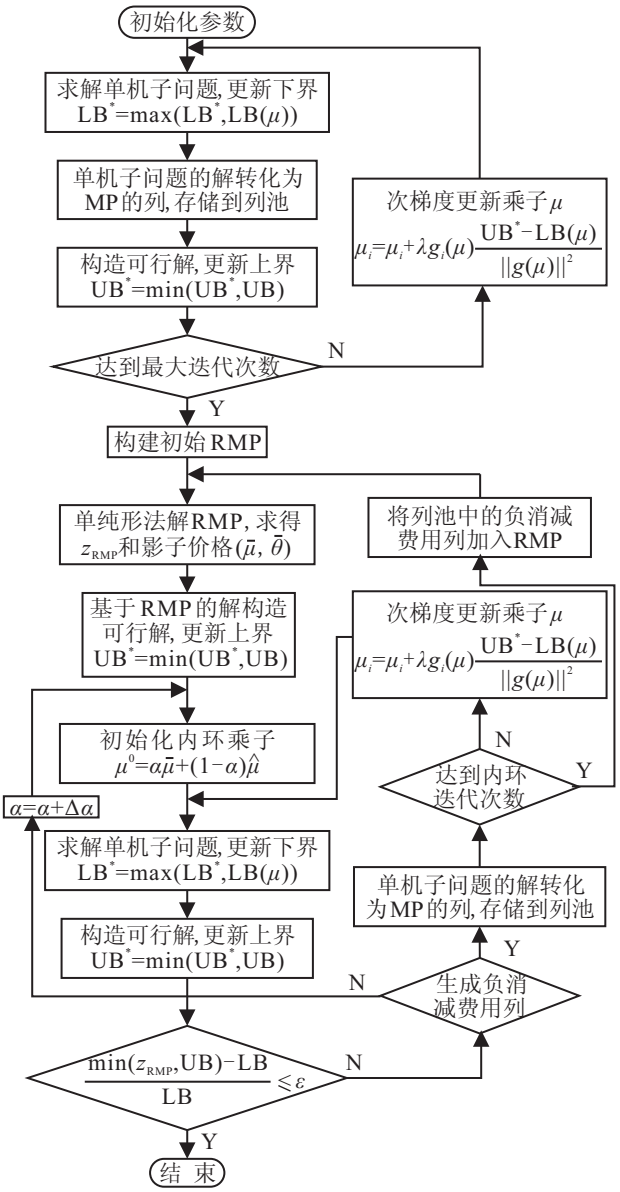

图 1　并行机实时调度问题的 LR & CG 算法流程

综上所述, 次梯度法和列生成算法各有优缺点, 且它们在迭代过程中涉及相同的子问题, 由此受到启发, 可以在次梯度法迭代寻找下界 LB^* 时, 将单机子问题的解转换为 RMP 的列并加入 RMP. 基于上述分析, 提出次梯度和列生成的混合策略, 称为 LR & CG 算法, 流程如图 1 所示.

LR & CG 算法包括 4 个关键要素:

1) 初始化 RMP.

执行固定代数的次梯度迭代, 由此得到当前最好拉格朗日乘子 $\hat{\mu}$ 和最好下界 $\mathrm{LB}^* = \mathrm{LB}(\hat{\mu})$. 在次梯度迭代过程中将机器 k 的单机子问题的解转换为 MP 的列并存储到列池 $\bar{\Omega}_k$. 如果将列池 $\bar{\Omega}_k(\forall k \in M)$ 中的所有列均加入, 则导致初始 RMP 的规模非常大, 在求解时耗费计算时间. 为了限制 RMP 规模, 假定乘子 $\hat{\mu}$ 为 LD 的近似最优解, 因此仅需要将列生成中满足下列条件的列加入 RMP:

$$c_\omega - \sum_{i\in N} a_{i\omega}\hat{\mu}_i - z_k(\hat{\mu}) \leqslant -\xi,\ k \in M,\ \omega \in \Omega_k, \quad (29)$$

其中 ξ 为一个较小值, 实验中取 $\xi = 0.01$.

2) 外环列生成.

采用单纯形算法求解 RMP, 得到最优值 z_{RMP} 和影子价格 $(\bar{\mu}, \bar{\theta})$. 当从外环转入内环时, 以影子价格 $\bar{\mu}$ 和最好拉格朗日乘子 $\hat{\mu}$ 的凸组合作为内环的初始乘子, 有

$$\mu^0 = \alpha\bar{\mu} + (1-\alpha)\hat{\mu},\ 0 < \alpha \leqslant 1. \quad (30)$$

定理 1　如果由式 (30) 得到的乘子 μ^0 是对偶可行解, 即基于乘子 μ^0 求得单机子问题的解对应的列的消减费用非负, 则 $\mathrm{LB}(\mu^0) \geqslant \mathrm{LB}(\hat{\mu}) + \alpha(z_{\mathrm{RMP}} - \mathrm{LB}(\hat{\mu}))$.

证明　令 $x^*_{0jk}, x^*_{j0k}, x^*_{ijk}, C^*_{jk}(k \in M, j \in N, i \in B_{jk})$ 为乘子是 μ^0 时单机子问题的最优解, 有

$$\begin{aligned}&\mathrm{LB}(\mu^0) = \mathrm{LB}(\alpha\bar{\mu} + (1-\alpha)\hat{\mu}) = \\ &\alpha\Big(\sum_{k\in M}\sum_{i\in N}\Big(w_iC^*_{ik} - \sum_{j\in A'_{ik}} x^{\omega*}_{ijk}\bar{\mu}_i\Big) + \sum_{i\in N}\bar{\mu}_i\Big) + \\ &(1-\alpha)\Big(\sum_{k\in M}\sum_{i\in N}\Big(w_iC^*_{ik} - \sum_{j\in A'_{ik}} x^*_{ijk}\hat{\mu}_i\Big) + \sum_{i\in N}\hat{\mu}_i\Big).\end{aligned} \quad (31)$$

由式 (16) 可知

$$\mathrm{LB}(\hat{\mu}) \leqslant \sum_{k\in M}\sum_{i\in N}\Big(w_iC^*_{ik} - \hat{\mu}_i\sum_{j\in A'_{ik}} x^*_{ijk}\Big) + \sum_{i\in N}\hat{\mu}_i. \quad (32)$$

因为 $x^*_{0jk}, x^*_{j0k}, x^*_{ijk}, C^*_{jk}(k \in M, j \in N, i \in B_{jk})$ 对应列的消减费用非负, 所以有

$$\sum_{i\in N} w_i C_{ik}^* - \sum_{i\in N}\sum_{j\in A'_{ik}} x_{ijk}^*\bar{\mu}_i - \bar{\theta}_k \geqslant 0,$$
$$\forall k \in M. \tag{33}$$

由式(31)～(33)可以得到

$$\mathrm{LB}(\mu^0) \geqslant a\Big(\sum_{k\in M}\bar{\theta}_k + \sum_{i\in N}\bar{\mu}_i\Big) + (1-\alpha)\mathrm{LB}(\hat{\mu}). \tag{34}$$

由线性规划对偶理论可知 $z_{\mathrm{RMP}} = \sum_{k\in M}\bar{\theta}_k + \sum_{i\in N}\bar{\mu}_i$, 因此有

$$\mathrm{LB}(\mu^0) \geqslant \mathrm{LB}(\hat{\mu}) + \alpha(\mathrm{z_{RMP}} - \mathrm{LB}(\hat{\mu})). \tag{35}$$

综上, 定理1得证. □

定理1表明采用式(30)初始化内环乘子有两个优点: 当单机子问题能够生产负消减费用列时, 添加到RMP将使 z_{RMP} 降低; 否则, 可以提升下界LB*. 因此, 在LR & CG算法中, 求解单机子问题不会作无用功, 不是降低 z_{RMP} 就是提升LB*, 由此实现算法的快速收敛.

3) 内环次梯度.

以 μ^0 为初始乘子, 执行少数几步次梯度迭代. 在此过程中, 如果找到更好的乘子, 则更新 $\hat{\mu}$ 和LB*, 另外还将单机子问题的解转换为MP的列并存储到列池. 若列池中存在负消减费用的列(影子价格为 $\bar{\mu}$), 则将负消减费用列加入RMP, 并转到外环求解RMP; 否则, 令因子 $\alpha = \alpha + \Delta\alpha$, 在新的因子下重新组合初始乘子 μ^0 并执行次梯度迭代. 当 $\alpha = 1$ 时, $\mu^0 = \bar{\mu}$, 若求解单机子问题仍不能生成负费用列, 则表明MP(LD)已达到最优, 终止算法. 因此, 在LR & CG算法中内环承担两个角色, 即下界求解器和列生成器. 同样, 外环也承担两个角色, RMP求解器和乘子调节器.

4) 构造可行解.

在外环中, 如果RMP的解为整数解, 即可得到原问题的可行解. 但在迭代过程中, 并不能保证每一步求得的RMP的解均为整数解. 在内环中, 由于松弛问题的解可能违反约束(2), 其解对原问题一般是不可行的. 因此, 需要利用当前信息构造可行解, 以获得原问题的上界.

在内环中, 基于 m 个单机子问题的解信息构造可行解. 定义 m 个单机子调度为 $(\omega_1^*, \omega_2^*, \cdots, \omega_m^*)$. 计算每个工件 i 被调度的次数 $\upsilon_i = \sum_{k\in M} a_{i\omega_k^*}$, 其中 $a_{i\omega_k^*}$ 为 ω_k^* 中工件 i 出现的次数. 对于任意工件 i, 如果 $\upsilon_i \geqslant 2$, 则在 $(\omega_1^*, \omega_2^*, \cdots, \omega_k^*)$ 中删除重复出现的 i 并保证 i 仅在某一个单机子调度 ω_k^* 中出现一次. 对于所有未调度 ($\upsilon_i = 0$) 的工件, 按 r_i 的非降序排列(相同 r_i 的工件按照 d_i 的非降序排列), 然后依次将未调度的工件插入 $(\omega_1^*, \omega_2^*, \cdots, \omega_k^*)$ 中的一个单机子调度, 确保满足时间窗约束并使目标函数改变量最小.

在外环中, 基于RMP的解信息来构造原问题的可行解. 定义 $\bar{\Omega}_k (k \in M)$ 为RMP生成的列, 计算工件 i 分配到机器 k 的比率 $\rho_{ik} = \sum_{\omega\in\bar{\Omega}_k} a_{i\omega}\lambda_k^\omega$, 将 $\rho_{ik} = 1$ 的工件 i 固定到机器 k 上. 将所有工件按 r_i 非降序排列(相同 r_i 按 d_i 非降序排列), 然后依次将工件 i 分配到 ρ_{ik} 最大的机器上, 如果工件 i 的所有非零 ρ_{ik} 均相等, 则将 i 分配到负荷最小的机器上. 对于任意机器 k 上的工件, 按 r_i 非降序排列(相同 r_i 按 d_i 非降序排列), 依次将工件 i 安排在满足时间槽约束且使目标函数改变最小的位置上.

4 实验结果

本文所提出的LR & CG算法使用VC++6.0语言编码, 在个人计算机(Intel Core(TM)2 Quad 2.83 GHz CPU和3.25 GB内存)上进行性能测试, 并以常规LR算法为benchmark算法, 对比两种算法的计算结果. LR & CG算法的停止准则为限制主问题的对偶间隙 $(z_{\mathrm{RMP}} - \mathrm{LB}^*)/\mathrm{LB}^*$ 小于 $1.0\mathrm{e}^{-5}$. 为了公平比较两种算法, LR算法在达到与LR & CG算法相同的计算时间时停止.

所有计算实例均基于文献[18]中的方法随机产生. 工件数 n 在{20, 40, 60, 80, 100}中选择, 机器数 m 在{2, 4, 6, 8, 10}中选择. 上述参数组合产生50种不同规模的问题, 对每一个规模问题, 随机产生10个不同实例, 共测试500个实例. 加工时间 p_{ik} 在[20, 50]满足均匀分布, 权重 w_i 在[1, 10]满足均匀分布. 时间窗 $[r_i, d_i]$ 的产生方法如下: 将 n 个工件随机安排到 m 台机器上, 令 t_i 为上述随机调度中工件 i 的开始时间, 令 $r_i = t_i - U[1, 60]$, $d_i = t_i + p_i k + U[1, 60]$. 为了防止 r_i 为负数, 调整 $r_i = r_i - \min\{r_j < 0 | j \in N\}$, $d_i = d_i - \min\{r_j < 0 | j \in N\}$.

表1分别从对偶间隙、上界和下界改进量、子问题迭代次数、计算时间等方面比较了LR & CG算法和LR算法的性能. 令 $\mathrm{UB_A}$ 和 $\mathrm{LB_A}$ 分别表示算法A的上界和下界, LR & CG的对偶间隙为

$$\mathrm{Gap_{LR\,\&CG}} = (\mathrm{UB_{LR\,\&CG}} - \mathrm{LB_{LR\,\&CG}})/\mathrm{LB_{LR\,\&CG}}.$$

LR的对偶间隙为

$$\mathrm{Gap_{LR}} = (\mathrm{UB_{LR}} - \mathrm{LB_{LR}})/\mathrm{LB_{LR}}.$$

LR & CG较LR的上、下界改进量分别为

$$\mathrm{Imp_{UB}} = (\mathrm{UB_{LR}} - \mathrm{UB_{LR\&CG}})/\mathrm{UB_{LR\&CG}},$$
$$\mathrm{Imp_{LB}} = (\mathrm{LB_{LR\&CG}} - \mathrm{LB_{LR}})/\mathrm{LB_{LR\&CG}}.$$

表1中每行是相同规模的10个实例的平均值, 由表1可以得到以下结论:

1) 在相同的计算时间下, 由LR & CG和LR得到的平均对偶间隙分别为2.55 %和3.77 %. LR & CG的

表 1　LR & CG 和 LR 算法的计算结果比较

问题规模 $n \times m$	对偶间隙/%		上界和下界改进量/%		子问题迭代次数		计算时间/s
	$Gap_{LR\&CG}$	Gap_{LR}	Imp_{LB}	Imp_{UB}	LR&CG	LR	
20×2	2.21	3.92	1.28	0.39	333	532	1.43
20×4	2.61	5.06	1.27	1.15	400	654	1.60
20×6	2.84	4.86	1.14	0.82	421	715	2.32
20×8	2.39	3.04	0.35	0.28	431	688	2.41
20×10	3.11	4.89	0.72	1.04	373	586	3.08
40×2	1.87	2.92	0.45	0.58	401	638	5.43
40×4	1.98	3.67	0.97	0.70	369	706	6.88
40×6	1.49	1.81	0.11	0.21	503	841	11.16
40×8	2.59	5.06	1.61	0.80	432	865	12.67
40×10	3.54	5.38	0.57	1.22	378	678	28.85
60×2	1.10	1.99	0.42	0.46	399	863	8.58
60×4	2.04	3.02	0.42	0.50	455	953	14.38
60×6	3.84	6.08	1.03	1.19	376	856	26.77
60×8	2.40	5.14	0.98	1.71	449	1024	21.27
60×10	3.28	5.20	0.85	1.00	439	1211	28.80
80×2	1.26	1.76	0.24	0.26	474	1517	27.09
80×4	1.71	2.22	0.23	0.28	524	1508	29.17
80×6	1.88	2.62	0.32	0.39	436	1608	44.53
80×8	2.77	3.47	0.36	0.32	443	1349	35.58
80×10	2.75	3.65	0.25	0.64	476	1370	73.90
100×2	1.02	1.20	0.05	0.13	574	2817	51.84
100×4	2.46	3.51	0.49	0.55	504	2234	45.18
100×6	2.24	2.58	0.05	0.27	573	2990	59.97
100×8	2.35	2.85	0.17	0.31	572	3269	76.48
100×10	2.99	3.45	0.16	0.29	653	3115	80.08
平均	**2.55**	**3.77**	**0.58**	**0.62**	**455**	**1344**	**27.98**

对偶间隙比 LR 平均降低了 1.22 %, 尤其是 LR 对偶间隙较大的实例, LR & CG 的对偶间隙降低得更多.

2) 对偶间隙的降低来源于下界和上界两方面的改进, 下界和上界的平均改进量分别为 0.58 % 和 0.62 %. LR & CG 算法能获得 LD 问题的精确解, LR 算法只能获得 LD 问题近似解, 因此下界改进是必然结果. 由于 LR & CG 的下界提供了更接近原问题最优解的信息, 基于这些信息构造的可行解也更接近于最优解, 从而改进了上界.

3) 对于所有实例, LR & CG 算法单机子问题迭代次数均小于 LR 算法, 这是因为两个算法总计算时间相同, 而 LR & CG 算法需要耗费部分计算时间来计算 RMP 和管理列, 所以用于单机求解子问题的时间变少, 迭代次数也随之减少.

4) 当工件数固定时, 随着机器数的增加, 计算时间增加. 这是由于机器数的增加导致每一次迭代需要求解更多的单机子问题. 当机器数固定时, 随着工件数的增加, 计算时间也增加, 这是由于工件数的增加导致单机子问题的动态规划的计算复杂度增加.

5) 对于所有测试实例, 提出的 LR & CG 算法能够在 100 s 内获得问题的近优解, 表明 LR & CG 算法对于求解并行机实时调度这类组合最优化问题具有一定潜力.

5　结　论

研究了目标函数为最小化总加权完成时间的并行机实时调度问题. 在探究拉格朗日松弛和列生成的内在联系的基础上, 提出一种融合两种算法的混合策略, 以充分发挥算法的各自优势, 实现求解过程的快速收敛. 同时还设计了基于次梯度的常规拉格朗日松弛算法作为 benchmark 测试混合算法的性能. 实验结果表明, 在相同的计算时间内, 混合算法获得的上界、下界和对偶间隙均优于常规拉格朗日松弛算法. 因此, 混合算法收敛速度更快, 性能更好. 本文所提出的混合算法还可以扩展到求解其他类型的并行机调度, 且可以从对偶的角度进一步探讨下界提升策略.

参考文献(References)

[1] Hoogeveen J A, Velde S L. Earliness-tardiness scheduling around almost equal due dates[J]. Informs J on Computing, 1997, 9(1): 92-99.

[2] De P, Ghosh J B, Wells C E. Due-dates assignment and early/tardy scheduling on identical parallel machines[J]. Naval Research Logistics, 1994, 41(1): 17-32.

[3] Cheng T C E, Chen Z L. Parallel-machine scheduling problems with earliness and tardiness penalties[J]. J of Operational Research Society, 1994, 45(6): 685-695.

[4] Arkin E M, Silverberg E B. Scheduling jobs with fixed start

and end times[J]. Discrete Applied Mathematics, 1987, 18(1): 1-8.

[5] Bouzina K I, Emmons H. Interval scheduling on identical machines[J]. J Global Optimization, 1996, 9(3/4): 379-393.

[6] Bruno J, Coffman E G, Sethi R. Scheduling independent tasks to reduce mean finishing time[J]. Communications of the ACM, 1974, 17(7): 382-387.

[7] Bar-Noy A, Guha S, Naor J S, et al. Approximating the throughput of multiple machines in real-time scheduling[J]. SIAM J on Computing, 2001; 31(2): 331-352.

[8] 霍满臣, 唐立新. 基于到达时间两台并行机上在线批调度[J]. 控制与决策, 2009, 24(12): 1826-1830.
(Huo M C, Tang L X. On-line batch scheduling with real time on two parallel machines[J]. Control and Decision, 2009, 24(12): 1826-1830.)

[9] 宫华, 唐立新. 并行机生产与具有等待时间限制的成批运输协调调度问题[J]. 控制与决策, 2011, 26(6): 921-924.
(Gong H, Tang L X. Scheduling production on parallel machines and batch delivery with limited waiting time constraint[J]. Control and Decision, 2011, 26(6): 921-924.)

[10] Sivrikaya F, Ulusoy G. Parallel machine scheduling with earliness and tardiness penalties[J]. Computers & Operations Research, 1999, 26(8): 773-787.

[11] Kaplan S, Rabadi G. Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times[J]. Computers & Industrial Engineering, 2012, 62(1): 276-285.

[12] Akker J M, Hoogeveen J A, Velde S L. Parallel machine scheduling by column generation[J]. Operations Research, 1999, 47(6): 862-872.

[13] Chen Z L, Lee C Y. Parallel machine scheduling with a common due window[J]. European J of Operational Research, 2002, 136(3): 512-527.

[14] Chen Z L, Powell W B. Exact algorithms for scheduling multiple families of jobs on parallel machines[J]. Naval Research Logistics, 2003, 50(7): 823-840.

[15] Kedad-Sidhoum S, Solis Y R, Sourd F. Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates[J]. European J of Operational Research, 2008, 189(3): 1305-1316.

[16] 罗守成, 陈峰, 唐国春. 平行机排序问题的列生成解法[J]. 系统科学与数学, 2008, 28(6): 739-746.
(Luo S C, Chen F, Tang G C. Column generation for solving parallel machine scheduling problem[J]. J of Systems Science and Complexity, 2008, 28(6): 739-746.)

[17] Rocha de Paula M, Mateus G R, Ravetti M G. A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times[J]. Computers & Operations Research, 2010, 37(5): 938-949.

[18] Gélinas S, Soumis F. A dynamic programming algorithm for single machine scheduling with ready times[J]. Operations Research, 1997, 69: 135-156.

(上接第828页)

[13] Omori K, Fujiwara Y, Maekawa S, et al. Evolutionary computation based on sexual selections driving asymmetrical mutations[J]. Trans of the Institute of Systems, Control and Information Engineers, 2002, 15(8): 422-429.

[14] Deb K, Agrawal R B. Simulated binary crossover for continuous search space[J]. Complex systems, 1995, 9(2): 115-148.

[15] Deb K, Agrawal S, Pratap A, et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization[J]. Lecture Notes in Computer Science, 2000, (1917): 849-858.

[16] Suganthan P N, Hansen N, Liang J J, et al. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization[D]. Singapore: School of Electrical and Electronic Engineering, Nanyang Technological University of Singapore, 2005.

[17] Blumer A, Ehrenfeucht A, Haussler D, et al. Occam's razor[J]. Information Processing Letters, 1987, 24(6): 377-380.

[18] Goh K S, Lim A, Rodrigues B. Sexual selection for genetic algorithms[J]. Artificial Intelligence Review, 2003, 19(2): 123-152.

[19] Miller B L, Goldberg D E. Genetic algorithms, tournament selection and the effects of noise[J]. Evolutionary Computation, 1996, 4(2): 113-131.