文章编号：1001-0920(2009)12-1826-05

基于到达时间两台并行机上在线批调度

霍满臣[1,2]，唐立新[2]

(1. 沈阳工程学院 数学教研室，沈阳 110136；2. 东北大学 流程工业综合自动化教育部重点实验室，沈阳 110004)

摘 要：考虑两台同构并行机上在线批调度问题.每个批具有不确定的到达时间，一旦机器可以利用，要在当前可以利用的批中选择出合适的批，并将其中的工件调度到机器上，且工件在加工过程中不允许中断.目标函数是使调度的最大完成时间最小.给出了一个批在线调度 RBLPT-算法，即选择当前批中加工时间之和最大的批按 LPT 规则调度.另外，利用反证法，对算法的最坏情况进行了分析.

关键词：最大完成时间；最坏情况比；同构并行机；最小反例；加工时间

中图分类号：TP301.6 **文献标识码**：A

On-line batch scheduling with real time on two parallel machines

HUO Man-chen[1,2]，*TANG Li-xin*[2]

(1. Department of Mathematical, Shenyang Engineering Institute, Shenyang 110136, China; 2. Key Laboratory of Integrated Automation for the Process Industry of Ministry of Education, Northeastern University, Shenyang 110004, China. Correspondent: HUO Man-chen, E-mail: huomanchen@163.com)

Abstract: On-line batch scheduling problem on two parallel machines is considered. Every batch has uncertain ready time. Once a machine is available, some batch is determined and the jobs in it are scheduled. Non-preemptive is permitted for processing jobs with objective to minimize makespan. An on-line batch scheduling RBLPT-algorithm is given, choose the batch with the maximum sum of processing time and then schedule by LPT-rule (the longest processing time first rule). By contradiction, the worst case is analyzed.

Key words: Makespan; Worst case ratio; Identical parallel machine; Minimum contrary example; Processing time

1 引 言

生产调度(Scheduling)广泛存在于生产和物流系统中，是典型的组合最优化问题[1].生产调度根据调度信息的完整性可分为离线调度和在线调度.离线调度是指在调度时刻工件的信息全部可利用；在线调度是指信息在调度开始执行时还不完全知道，调度的信息是随着调度的执行进程而逐渐给出的[2].在以往的研究中，人们主要关注离线调度问题，而实际生产中经常出现调度的信息不能全部给出的情况.例如，钢铁企业的钢锭加工过程是一个动态的过程，在调度决策开始时，对钢锭的全部信息并不知道，其信息是随着轧制的进程逐渐给出的[3-5].由于在线调度问题的广泛存在性，近年来已成为调度领域的研究热点.

经典在线调度问题主要有两种形式：1)工件一个接一个地到达，形成了一个工件列表；2)工件按不确定的到达时间到达.这两种形式的共同特点是工件单独到达，等待机器加工[6].本文将工件的到达方式推广为批到达方式.现考虑它的一个简单形式，讨论在两台同构并行机上的在线批调度问题[7].这里的批工件具有不确定的到达时间[8]，即每批工件的到达时间是未知的.将工件调度到两台同构并行机上，且只有当前批中的所有工件全部调度完时，才可以调度下一批中的工件.目标是使最大完成时间最小.文中给出了一个在线批启发式算法：RBLPT-算法(按 LPT 规则制定的具有到达时间的在线批调度算法)，要求在每一个批中的工件均按 LPT(the longest processing time first rule)规则调度[9].研究表明，该算法的最坏情况比为 3/2.

2 在线批调度

收稿日期：2008-11-28；**修回日期**：2009-04-09.

基金项目：高等学校学科创新引智计划项目(B08015)；国家杰出青年科学基金项目(70425003)；国家自然科学基金项目(60674084)；辽宁省教育厅项目(20060589).

作者简介：霍满臣(1963—)，男，辽宁海城人，教授，博士，从事生产调试与物流优化管理等研究；唐立新(1966—)，男，黑龙江兰西人，教授，博士生导师，从事物流优化与控制等研究.

本文研究在两台同构并行机上工件具有到达时间的在线批调度问题. 其描述如下：有两台同构并行机，记为 M_1 与 M_2. 每台机器一次只可以加工一个工件且在加工过程中不可以中断，即只有当机器上的工件加工完成时才可以加工其余等待加工的工件. 每批中的工件具有到达时间且到达时间是未知的. 每批中所有工件的特征信息（如每个工件的加工时间）只有在该批到达时才可以得到. 在任时刻，当已加工批中工件全部进行加工且某台机器可以利用的情况下，才可以对当前已到达批中的工件进行调度. 问题的目标函数是使调度的最大完成时间最小.

由于两台同构并行机上的离线调度问题 $P_2 \| C_{\max}$ 是 NP-hard 的[1]，在两台同构并行机上的在线批调度问题至少与 $P_2 \| C_{\max}$ 一样难. 本文所研究的两台同构并行机上工件具有到达时间的在线批调度问题显然是 NP-hard 的，因此对该问题建立一个在线批调度启发式算法[10]，记为 A，并对算法性能进行分析. 设 I 为问题的任一实例，$A(I)$ 表示由在线算法 A 对实例 I 所产生的调度 σ 的最大完成时间，且 $\mathrm{OPT}(I)$ 表示在离线调度下对实例 I 产生最优调度 π 的最大完成时间. 则算法 A 的性能由 $A(I)$ 与 $\mathrm{OPT}(I)$ 比值的上确界值大小来度量. 该比值的上确界用 R^A 表示，称为算法的最坏情况比，或算法的竞争率[11].

3　批工件具有到达时间的 RBLPT- 算法

设每批工件具有一个不确定的到达时间. 在任一时刻，当某台机器可利用时，将当前已到达批中全部工件加工时间之和最大的批调度到机器上，且将每批中的工件按 LPT 规则调度到两台同构并行机器上. 当这批工件完全加工完成后，才可以加工其后一批中的工件.

本文引入下列符号（符号中 $i = 1,2,\cdots,n$；$j = 1,2,\cdots,b_i$）：

B_i：第 i 批被调度工件集合；

B_l：最后加工完成的一批工件集合；

b_i：批 B_i 中的所含工件数量；

J_{ij}：第 i 个被调度的批 B_i 中第 j 个工件；

p_{ij}：第 i 个被调度的批 B_i 中第 j 个工件的加工时间；

r_i：批 B_i 的到达时间；

S_i：批 B_i 的开始加工时间；

S_{i1}：第 i 个被调度批 B_i 在第 1 台机器 M_1 上的开始加工时间；

S_{i2}：第 i 个被调度批 B_i 在第 2 台机器 M_2 上的开始加工时间；

P_{i1}：第 i 个被调度批 B_i 在第 1 台机器 M_1 上的加工时间之和；

P_{i2}：第 i 个被调度批 B_i 在第 2 台机器 M_2 上的加工时间之和；

C_{i1}：第 i 个被调度批 B_i 在机器 M_1 上的完成时间；

C_{i2}：第 i 个被调度批 B_i 在机器 M_2 上的完成时间；

λ_1：在调度 σ 下最后加工完成的这批工件在机器 M_1 上所有工件加工时间之和；

λ_2：在调度 σ 下最后加工完成的这批工件在机器 M_2 上所有工件加工时间之和；

$A(I)$：算法 A 对实例 I 产生调度 σ 的最大完成时间；

$\mathrm{OPT}(I)$：最优算法对实例 I 产生的调度 π 的最大完成时间.

3.1　RBLPT- 算法

不妨设有 n 批工件 B_i（$i = 1,2,\cdots,n$），它们中的工件的加工时间随批的到达而定.

Step1：设 $i = 1$.

Step2：当 $i \leqslant n$ 时，将当前时刻每批中工件的加工时间累加起来，加工时间和最大的批确定为当前时刻要调度的批，记为 B_i. 当 $i = n + 1$ 时，转 Step8.

Step3：批 B_{i-1} 中的工件在机器 M_1 上全部加工完成时，若 $C_{i-1,1} \leqslant r_i$，则设 $S_{i1} = r_i$；若 $C_{i-1,1} > r_i$，则设 $S_{i1} = C_{i-1,1}$. 批 B_{i-1} 中的工件在机器 M_2 上全部加工完成时，若 $C_{i-1,2} \leqslant r_i$，则设 $S_{i2} = r_i$；若 $C_{i-1,2} > r_i$，则设 $S_{i2} = C_{i-1,2}$.

Step4：将批 B_i 中工件按加工时间大小非降次序排序，不妨设为 $P_{i1} \geqslant P_{i2} \geqslant \cdots \geqslant P_{ib_i}$.

Step5：在时刻 $\min\{S_{i1}, S_{i2}\}$，按 LPT 规则调度第 i 批 B_i 中的工件，并得出该批工件在机器 M_1，M_2 上的加工时间 P_{i1}，P_{i2}.

Step6：计算 $C_{i1} = C_{i-1,1} + P_{i1}$ 及 $C_{i2} = C_{i-1,2} + P_{i2}$.

Step7：$i = i + 1$，转 Step2.

Step8：调度结束，且设调度的最大完成时间 $C_{\max} = \max\{C_{n1}, C_{n2}\}$.

3.2　算法举例

设有 3 个批 B_1，B_2，B_3，它们的到达时间分别为 $r_1 = 0$，$r_2 = 5$，$r_3 = 5$. 批 B_1 中有 5 个工件，其处理时间分别为 $P_{11} = 2$，$P_{12} = 1$，$P_{13} = 3$，$P_{14} = 2$，$P_{15} = 1$；批 B_2 中有 4 个工件，其处理时间分别为 $P_{21} = 2$，$P_{22} = 3$，$P_{23} = 1$，$P_{24} = 4$；批 B_3 中有 3 个工件，其处理时间分别为 $P_{11} = 2$，$P_{12} = 4$，$P_{13} = 3$. 由 RBLPT- 算法在时刻 $r_1 = 0$ 利用 LPT 规则对 B_1 中

的工件进行调度. 在时刻 $r_2 = r_3 = 5$ 两个批同时到达,其中 B_2 中所有工件处理时间之和为 $P_{21}+P_{22}=10$,B_3 中所有工件处理时间之和为 $P_{31}+P_{32}=9$. 而按算法取批中处理时间之和最大的批先按 LPT- 规则进行调度. 因为 $P_{21}+P_{22}>P_{31}+P_{32}$,所以在时刻 $r_2=r_3=5$,要先对批 B_2 中工件按 LPT 规则进行调度. 对于批 B_3,当机器可利用时,对批 B_3 中的工件按 LPT- 规则进行调度. 调度的最大完成时间为 16,而相应的最优算法产生的调度最大完成时间为 15. 从而说明 RBLPT- 算法不是最优算法,因此有必要对算法的最坏情况进行分析.

4 批具有到达时间的算法最坏情况分析

设批 B_i 有 n_i(n_i 为任意的自然数)个工件 J_{i1},J_{i2},…,J_{in_i} 且批 B_i 的到达时间为 r_i,其中工件 J_{ij} 随着批的到达,它们的加工时间 p_{ij}($i=1,2,\cdots,n;j=1,2,\cdots,J_{in_i}$)便给定. 不失一般性,假定当某两个工件的完成时间相等时,将下一个工件(如果还有的话)分配到两台机器中具有最大加工时间工件的机器上.

现在利用反证法对算法的最坏情况进行分析. 假设这一算法的最坏情况比超过 3/2,这样就一定存在一类实例,算法对该类实例所产生的最大完成时间大于最优调度产生的最大完成时间的 3/2 倍. 假设 BL 这类实例中含有最少数目批工件的一个实例,σ 为由在线 RBLPT- 算法对实例 BL 产生的调度,π 为最优算法对实例 BL 产生的调度. 由假设有

$$A(\mathrm{BL})>\frac{3}{2}\mathrm{OPT}(\mathrm{BL}).$$

引理 1 在线的具有到达时间的 RBLPT- 算法对实例 BL 产生的调度 σ 中,在时间区间 $[0,C_{\max}(\sigma)]$ 的任一子时间区间内,至少有一台机器不空闲.

证明 如果 RBLPT- 算法对实例 BL 产生的调度 σ 存在一个公共的空闲时间区间 $[t_1,t_2]\subseteq[0,C_{\max}(\sigma)](t_1\leqslant t_2)$,即在这一区间上所有机器都没有工件可加工,则至少存在一个批,其中的所有工件在时刻 t_1 前完成加工. 由 RBLPT- 算法知,调度在这个公共空闲时间区间 $[t_1,t_2]$ 之后的所有批中工件的到达时间一定不小于 t_2. 如果去掉所有调度在时间区间 $[t_1,t_2]$ 之前的批工件,则可以得到一个新的更小实例 BL′. 对于这一实例 BL′,由 RBLPT- 算法产生的调度设为 σ',则调度的最大完成时间 $A(\mathrm{BL}')$ 保持不变,即 $A(\mathrm{BL}')=A(\mathrm{BL})$. 而相应的离线状态下最优调度的最大完成时间 $\mathrm{OPT}(\mathrm{BL}')$ 不会增大,而且还有可能减小,即 $\mathrm{OPT}(\mathrm{BL}')\leqslant\mathrm{OPT}(\mathrm{BL})$. 因为

$$A(\mathrm{BL})>\frac{3}{2}\mathrm{OPT}(\mathrm{BL}),$$

所以可以推出新工件数目较少的实例的两个算法的调度最大完成时间之间具有如下同样的不等关系:

$$A(\mathrm{BL}')=A(\mathrm{BL})>\frac{3}{2}\mathrm{OPT}(\mathrm{BL})\geqslant\frac{3}{2}\mathrm{OPT}(\mathrm{BL}').$$

即新的实例 BL′ 在调度为 σ' 下满足 $A(\mathrm{BL}')>1.5\,\mathrm{OPT}(\mathrm{BL}')$. 这样新的 BL′ 是一个含有更小数目工件的实例,这与假设中 BL 是最小批工件实例相矛盾. 所以,对于最小实例不存在这样的空闲时间. 另一方面,如果存在这样的空闲时间区间 $[t_1,t_2]$,则调度的开始时间应为 t_2,即 $r_1=t_2$. 所有批都应在这一时间区间后到达. 现将实例 BL 中所有批工件的到达时间减少 r_1,则对应的算法产生的调度及最优调度的最大完成时间都减少相同的量 r_1. 而

$$\frac{C_{\max}(\sigma)-r_1}{C_{\max}(\pi)-r_1}>\frac{C_{\max}(\sigma)}{C_{\max}(\pi)}>\frac{3}{2},$$

从而改变以后所得到的新实例还是一个含有最小数目的实例. 这样,不失一般性,可以认为在时间区间 $[0,A(\mathrm{BL})]$ 的任一子时间区间上,至少有一台机器不空闲. □

引理 2 如果实例 BL 是由 RBLPT- 算法对所产生的调度 σ 满足

$$A(\mathrm{BL})>\frac{3}{2}\mathrm{OPT}(\mathrm{BL}) \tag{1}$$

的最小实例,则实例 BL 中最后到达的批最后完成加工.

证明 假设最后到达的批 B_l 不是最后完成加工的,则可将这一批中的工件从最小实例 BL 中删除,从而得到一个新的更小批工件的实例 BL′,对应的调度记为 σ',从而新批工件列 BL′ 中批的数量少于原批工件列 BL 批的数量. 显然,由批在线 RBLPT- 算法产生的在线调度的最大完成时间不变,即 $A(\mathrm{BL})=A(\mathrm{BL}')$,而最优离线调度的最大完成时间由于批的数量的减少可能减小,也可能不发生变化,即 $\mathrm{OPT}(\mathrm{BL})\leqslant\mathrm{OPT}(\mathrm{BL}')$. 从而有

$$\frac{A(\mathrm{BL}')}{\mathrm{OPT}(\mathrm{BL}')}\geqslant\frac{A(\mathrm{BL})}{\mathrm{OPT}(\mathrm{BL})}>\frac{3}{2}.$$

这样新产生了一个满足条件

$$\frac{A(\mathrm{BL}')}{\mathrm{OPT}(\mathrm{BL}')}>\frac{3}{2}$$

的更小批工件实例 BL(批的数量小于原来批的数量),这与实例 BL 满足 $\frac{A(\mathrm{BL})}{\mathrm{OPT}(\mathrm{BL})}>\frac{3}{2}$ 的最小批工件实例相矛盾. 因此对于最小批工件实例 BL,其最后到达的批 B_l 将最后完成加工. □

定理 1 对于调度 σ,最后一批完成加工工件的批开始加工时间 $S_l(\sigma)$ 与批到达时间 r_l 之差满足

$$S_l(\sigma)-r_l>\frac{\mathrm{OPT}(\mathrm{BL})}{2}-\frac{|\lambda_2-\lambda_1|}{2}, \tag{2}$$

其中 $|\lambda_2-\lambda_1|$ 为最后加工完成的批工件在两台机器上加工时间和之差的绝对值.

证明　由假设,对于 RBLPT- 算法产生的调度 σ 的最大完成时间与离线最优算法产生的调度 π 的最大完成时间之间有如下关系:

$$A(\mathrm{BL})>\frac{3}{2}\mathrm{OPT}(\mathrm{BL}).$$

而离线情况下的最优算法产生的调度 σ 的最大完成时间,不小于最后到达的这一批工件的到达时间与该批中所有工件的加工时间在两台同构机上的平均加工时间之和,即

$$\mathrm{OPT}(\mathrm{BL})\geqslant r_l+\frac{1}{2}\sum_{j=1}^{k}p_{lj}.$$

RBLPT- 算法产生的调度 σ 的最大完成时间 $A(\mathrm{BL})$ 与这批工件的开始加工时间 $S_l(\sigma)$ 有关,即完成时间不小于最后这批工件的开始加工时间与这批工件的所有工件完成时间在两台机器上平均加工时间之和,即

$$A(\mathrm{BL})\geqslant S_l(\sigma)+\frac{1}{2}\sum_{j=1}^{k}p_{lj}.$$

设 $\lambda_1(\lambda_2)$ 为在调度 σ 下最后完成加工这批工件在机器 $M_1(M_2)$ 上所有工件加工时间之和. 不妨设调度 σ 在机器 M_1 上最先开始加工. 设 S_{li} 为最后一批工件在机器 M_i 上的开始加工时间($i=1,2$),两台机器对最后一批工件的开始加工时间之差的绝对值为 $d=|S_{l1}-S_{l2}|$,而 $S_{l1}=S_l(\sigma)\leqslant S_{l2}$,所以两台机器上工件的开始加工时间之差的绝对值可表示为

$$d=S_{l2}-S_{l1}=S_{l2}-S(\sigma).$$

当最后完成加工工件的机器为 M_1 时,即这台机器对最后被调度的这批工件的开始加工时间为 $S_l(\sigma)$ 时,有 $A(\mathrm{BL})=S_l(\sigma)+\lambda_l$,此时有 $|\lambda_1-\lambda_2|\geqslant d$. 于是,最后这批工件的开始加工时间与到达时间之差为

$$\begin{aligned}&S_l(\sigma)-r_l=A(\mathrm{BL})-\lambda_1-r_l\geqslant\\&A(\mathrm{BL})-\lambda_1-\Big(\mathrm{OPT}(\mathrm{BL})-\frac{1}{2}\sum_{j=1}^{k}p_{lj}\Big)>\\&\frac{3}{2}\mathrm{OPT}(\mathrm{BL})-\mathrm{OPT}(\mathrm{BL})+\frac{\lambda_1+\lambda_2}{2}-\lambda_1=\\&\frac{1}{2}\mathrm{OPT}(\mathrm{BL})+\frac{\lambda_2-\lambda_1}{2}=\\&\frac{1}{2}\mathrm{OPT}(\mathrm{BL})-\frac{|\lambda_2-\lambda_1|}{2}.\end{aligned}$$

当最后完成加工工件的机器为 M_2 时,即这台机器对最后到达这批工件的开始加工时间不超过 $S_l(\sigma)$ 时,有

$$A(\mathrm{BL})=S_2(\sigma)+\lambda_2\geqslant S_l(\sigma)+\lambda_2.$$

其中

$$S_2(\sigma)=S_l(\sigma)+d,\ |\lambda_1-\lambda_2|\geqslant d.$$

从而最后这批工件的开始加工时间与到达时间之差为

$$\begin{aligned}&S_l(\sigma)-r_l=A(\mathrm{BL})-\lambda_2-r_l\geqslant\\&A(\mathrm{BL})-\lambda_2-\Big(\mathrm{OPT}(\mathrm{BL})-\frac{1}{2}\sum_{j=1}^{k}p_{lj}\Big)>\\&\frac{3}{2}\mathrm{OPT}(\mathrm{BL})-\mathrm{OPT}(\mathrm{BL})+\frac{\lambda_1+\lambda_2}{2}-\lambda_2=\\&\frac{1}{2}\mathrm{OPT}(\mathrm{BL})-\frac{\lambda_2-\lambda_1}{2}\geqslant\\&\frac{1}{2}\mathrm{OPT}(\mathrm{BL})-\frac{|\lambda_2-\lambda_1|}{2}.\end{aligned}$$

设时间区间 $[t_s,t_f]$ 为最后一个机器空闲时间段,则最后被调度的批的到达时间 r_l 一定大于 t_f;否则,该批中的工件就可以被调度到在这一时间段上空闲的机器上. □

定理 2　最后完成加工批中的工件在两台机器上完工时间满足

$$\max\{\lambda_1,\lambda_2\}\leqslant\frac{\mathrm{OPT}(\mathrm{BL})}{2}-\frac{t_f}{4}+\frac{\lambda_1+\lambda_2}{2}.\quad(3)$$

证明　因为 $t_f\leqslant r_l$,且由引理 2 有

$$S_l(\sigma)-r_l>\frac{\mathrm{OPT}(\mathrm{BL})}{2}-\frac{|\lambda_2-\lambda_1|}{2}>0.$$

从而可知最后被调度的批 B_l 到达后并没有立即被调度,且在其前面必有在 t_f 以后到达加工的批工件,所以有

$$\begin{aligned}&\mathrm{OPT}(\mathrm{BL})\geqslant t_f+|\lambda_1-\lambda_2|>\frac{t_f}{2}+|\lambda_1-\lambda_2|,\\&\mathrm{OPT}(\mathrm{BL})-\frac{t_f}{2}>|\lambda_1-\lambda_2|=\\&2\max\{\lambda_1,\lambda_2\}-(\lambda_1+\lambda_2),\\&2\max\{\lambda_1,\lambda_2\}<\mathrm{OPT}(\mathrm{BL})-\frac{t_f}{2}+(\lambda_1+\lambda_2),\\&\max\{\lambda_1,\lambda_2\}\leqslant\frac{\mathrm{OPT}(\mathrm{BL})}{2}-\frac{t_f}{4}+\frac{\lambda_1+\lambda_2}{2}.\end{aligned}$$ □

引理 3　在假设下 RBLPT- 算法产生的调度中,机器存在空闲时间.

证明　若算法所产生的调度中机器没有空闲时间,则 $S_l(\sigma)$ 显然为最优算法所产生的调度的最大完成时间的一个下界,即 $\mathrm{OPT}(\mathrm{BL})\geqslant S_l(\sigma)$. 从而有

$$\begin{aligned}&2\max\{\lambda_1,\lambda_2\}<\mathrm{OPT}(\mathrm{BL}),\\&C_{\max}(\sigma)\leqslant S_l(\sigma)+\max\{\lambda_2,\lambda_2\}\leqslant\\&\mathrm{OPT}(\mathrm{BL})+\frac{1}{2}\mathrm{OPT}(\mathrm{BL})=\\&\frac{3}{2}\mathrm{OPT}(\mathrm{BL}).\end{aligned}$$

这与假设 $A(\mathrm{BL}) > \frac{3}{2}\mathrm{OPT}(\mathrm{BL})$ 相矛盾. 因此在假设下 RBLPT- 算法产生的调度中机器存在空闲时间. □

定理 3 RBLPT- 算法产生的调度的开始时间 $S_l(\sigma)$ 满足下列关系:

$$S_l(\sigma) \leqslant \mathrm{OPT}(\mathrm{BL}) - \frac{\lambda_1+\lambda_2}{2} + \frac{t_f}{4}. \tag{4}$$

证明

$$\mathrm{OPT}(\mathrm{BL}) \geqslant t_f + (S_l(\sigma) - t_f) + \frac{\lambda_1+\lambda_2}{2} - \frac{S_j - r_j}{2} = S_l(\sigma) + \frac{\lambda_1+\lambda_2}{2} - \frac{S_j - r_j}{2},$$

$$\mathrm{OPT}(\mathrm{BL}) \geqslant (S_l(\sigma) - t_f) + (S_j - r_j) + \frac{\lambda_1+\lambda_2}{2} + \frac{t_f - (S_j - r_j)}{2} = S_l(\sigma) + \frac{\lambda_1+\lambda_2}{2} - \frac{t_f}{2} + \frac{S_j - r_j}{2}.$$

将上面两式相加,得

$$2\mathrm{OPT}(\mathrm{BL}) \geqslant 2S_l(\sigma) + (\lambda_1+\lambda_2) - \frac{t_f}{2},$$

从而有

$$S_l(\sigma) \leqslant \mathrm{OPT}(\mathrm{BL}) - \frac{\lambda_1+\lambda_2}{2} + \frac{t_f}{4}. \qquad \square$$

定理 4 RBLPT- 算法的最坏情况比不大于 3/2.

证明 由前面的定理,因为

$$S_l(\sigma) \leqslant \mathrm{OPT}(\mathrm{BL}) - \frac{\lambda_1+\lambda_2}{2} + \frac{t_f}{4},$$

$$\max\{\lambda_1,\lambda_2\} \leqslant \frac{\mathrm{OPT}(\mathrm{BL})}{2} - \frac{t_f}{4} + \frac{\lambda_1+\lambda_2}{2}.$$

所以有

$$A(\mathrm{BL}) \leqslant S_l(\sigma) + \max\{\lambda_1,\lambda_2\} \leqslant \mathrm{OPT}(\mathrm{BL}) - \frac{\lambda_1+\lambda_2}{2} + \frac{t_f}{4} + \frac{\mathrm{OPT}(\mathrm{BL})}{2} - \frac{t_f}{4} + \frac{\lambda_1+\lambda_2}{2} = \frac{3}{2}\mathrm{OPT}(\mathrm{BL}).$$

这与假设矛盾,因此,算法的最坏情况比应不大于 3/2. □

5 实验与数值计算

定义近似算法 RBLPT 求得的目标函数值为 $A(\mathrm{BL})$,相应离线最优算法的下界值为 $\mathrm{OPT}(\mathrm{BL})$.

算法是由 C 语言编程,在 Pentium-Ⅳ 的 PC 机上运行, 操作系统是 Windows XP,CPU 是 2.40GHz. 在数值计算中,对于 $n \in \{10,30,60,70,80,90,100\}$, $k \in \{2,4,6,8,60,80,100\}$ 的每种组合,RBLPT 算法产生的 $A(\mathrm{BL})$ 有 5 个算例,n 表示每个算例中批的数量. 每种情况作 50 次随机模拟,其中批工件的到达时间为 r_i,工件的基本处理时间 p_i 分别在[1,100],[1,100],[0,1] 上服从均匀分布. 表 1 给出了算法相应最坏情况比值在各个比值区间上模拟出现的次数,其中比值区间被分成[1.0,1.1), [1.1,1.2), [1.2,1.3), [1.3,1.4), [1.4,1.5].

表 1 实验与数值计算结果

比值区间	n				
	10	30	60	80	100
[1.0,1.1)	42	43	48	47	48
[1.1,1.2)	4	6	1	1	1
[1.2,1.3)	3	0	0	1	1
[1.3,1.4)	0	1	0	0	0
[1.4,1.5]	1	0	1	0	0
总计	50	50	50	50	50

基于表 1 中的解,可得如下分析结果:

1) 当批中工件数量较多时,算法的最坏情况比大多数在区间[1.0,1.1) 中,表明算法性能较好;

2) 因为启发式算法对于批内工件较多问题的改进比批内工件少的改进更大,所以减小的最大完成时间更加明显.

6 结 论

本文研究了一个具有到达时间的在线批调度问题. 通过批调度与在线调度相结合,提出了一种在线批调度算法:RBLPT-算法. 当机器可利用时,在当前已到达的批中取加工时间和最大的批为要调度的批工件,且每批中的工件按 LPT-规则进行调度. 文中利用反证法,通过对最后到达批的加工时间的特征分析,及最优算法的最大完成时间与算法产生的调度中最后到达批的开始时间的关系,证明了算法的最坏情况比为 3/2. 在实际中存在大量的本文提出的在线批调度问题,本文从理论上为这类问题提出了一个最坏情况比为 3/2 的算法,从而为实际生产中合理优化生产奠定了理论基础.

参考文献(References)

[1] Pinedo. 调度:原理、算法和系统[M]. 第 2 版. 北京:清华大学出版社, 2005.
(Pinedo. Scheduling: Theory, algorithms and systems [M]. 2nd ed. Bejing: Tsinghua University Press, 2005.)

[2] Sgall J. On-line scheduling, online algorithms: The state of the art[J]. Lecture Notes in Computer Science, 1998, 1442(5): 196-231. (下转第 1835 页)