运筹学与控制论

DOI :CNKI :50-1165/N. 20120314. 1927. 004

带有释放时间的半连续型批处理机调度问题*

王松丽，赵玉芳，崔苗苗
(沈阳师范大学 数学与系统科学学院，沈阳 110034)

摘要 :半连续批处理机调度问题,是从钢铁工业加热炉对管坯的加热过程中提炼出来的。工件按批加工,同一批中工件的加工时间等于此批中工件的最大加工时间,且工件必须按周期一个紧挨着一个进入、离开处理机。批处理机的容量为 C,即最多可同时加工 C 个工件,批的容量为批中工件的个数,批的处理时间与批中工件的加工时间、批处理的容量和批的容量有关。本文研究释放时间与加工时间一致时,对于目标函数为最大完工时间问题,即时间表长问题,分析其最优解的性质,从而将问题转化为工件按释放时间非减顺序排列后,对工件进行分批,使得最大完工时间最小。在此基础上给出了一个复杂性为 $O(n^2)$ 的动态规划算法,证明了这个算法的最优性,并用数值例子进一步说明了算法的计算过程。

关键词 :加热炉调度 ;半连续批 ;计算复杂性 ;动态规划算法
中图分类号 :O223　　文献标志码 :A　　文章编号 :1672-6693(2012)02-0016-08

文献[1]从钢铁工业加热炉对管坯的加热过程中提出的一种半连续型批处理机调度问题,其主要特征为工件按批加工,处理机的容量是指批处理机在同一时刻可同时加工工件的最大数,批的容量是指在同一批中按周期连续加工的工件数,由于处理机中工件是连续进、连续出的,所以批的容量可以是无限的,也就是说批处理机的容量与批的容量无关。同一批工件中最大的加工时间为此批工件的加工时间,批处理机的时间是从该批中的第一个工件进入到最后一个工件离开机器所用的时间,因此批的加工时间取决于批中工件的个数即批的容量、工件的最大加工时间及批处理机的容量。本文研究的是工件释放时间与加工时间一致时,极小化最大完工时间的单机半连续型批处理机调度问题,讨论其最优解的性质,并给出复杂性为 $O(n^2)$ 的动态规划算法,证明了算法的最优性,用数值例子进一步说明了算法的计算过程。数学模型具体描述如下。

设有 n 个相互独立工件 $J_j(j=1,2,\cdots,n)$ 在一台批处理机上加工,加工时间为 $p_j>0$,释放时间为 $r_j\geqslant 0$,且 r_j 与 p_j 是一致的,即 $r_i\geqslant r_j$ 时有 $p_i\geqslant p_j$,记为 $\mathrm{agr}(r_j,p_j)$。这些工件可以任意分成若干个批,记为 $B_1,B_2,\cdots,B_l$。批处理机的容量为 C,也就是说处理机最多可同时加工 C 个工件;批 B_k 的大小为批 B_k 中工件的个数(即批 B_k 的容量),记为 $|B_k|$。在加工过程中,同一批中工件的加工时间等于此批中所有工件加工时间的最大者,称其为批基本加工时间,记为 $p_{(k)}$,则 $p_{(k)}=\max_{J_j\in B_k}\{p_j\}$。工件必须按周期一个紧挨着一个进入、离开处理机,批的加工时间为从该批中的第一个工件进入机器,到最后一个工件离开机器所用的时间,记为 $p^{(k)}$,$p^{(k)}=p_{(k)}\left(1+\frac{|B_k|-1}{C}\right)$。只有当这批工件加工完之后才可以加工下一批工件。工件释放时间和加工时间一致的极小化最大完工时间的单机半连续型批处理机调度问题,用三参数表示法记为

$$1\,|\,c\text{-}batch,C,\mathrm{agr}(r_i,p_i)\,|\,C_{\max}$$

批调度问题一般分为传统的批处理机调度问题和新型的批处理机调度问题。而传统的批处理机调

* 收稿日期 :2011-07-16　修回日期 :2011-12-01　网络出版时间 :2012-03-14 19 :27 :00
资助项目 :国家 863 高技术发展计划(No. 2006AA04Z174);国家自然科学基金项目(No. 60674084);辽宁省教育厅高等学校科研项目(No. 2008Z192)
作者简介 :王松丽,女,硕士研究生,研究方向为组合最优化;通讯作者 :赵玉芳,E-mail :yfzhao2004@163. com
网络出版地址 :http ://www. cnki. net/kcms/detail/50. 1165. N. 20120314. 1927. 201202. 16_004. html

度问题一般都是批进批出[2],也就是同一批中工件同时开始加工、同时结束,主要分为两类:并行和串行批处理机调度问题。对于并行批处理机调度模型[3],批的加工时间等于此批中所有工件的加工时间的最大者;串行批处理机调度模型,批的加工时间等于此批中所有工件的加工时间之和。文献[3-8]等对传统的批处理机调度问题进行研究。其中对于批的内容量 B 是有限的并行批处理机的调度问题 $1|r_j,B|C_{\max}$,Lee 和 Uzsoy[4] 给出伪多项式时间的动态规划算法;Liu 和 Yu[5] 证明了即使批容量 $B=2$,上述问题也是一般意义 NP-难的;当批容量 B 无限制时,对于时间表问题,文献[6]证明了工件的到达时间和工期,或到达时间和加工时间一致单调时都是多项式时间可解的;当加工时间与工期一致单调时,该问题是 NP 困难的。

新型批处理机调度问题即半连续型批调度问题,是从钢铁工业中加热炉对管坯的加热过程中提出的。在钢铁企业生产中,加热炉只是多道工序的一个环节,其原料是由上游工序供给的,而上游工序的供料是动态的,但可以根据上道工序的生产预测出到达时间,即工件的到达时间是已知的。从加热炉的工作过程可以看出,由于管坯是通过步进梁的周期运动进入加热炉进行加热的,这样对于带有释放时间的管坯,若要进入加热炉加热,不仅工件要具有可利用性,同时必须在传输工具——步进梁可利用时才能实现。文献[1]研究了目标函数为极小化最大完工时间的调度问题,证明其是多项式时间可解的,并给出动态规划算法。文献[8]研究了目标函数是总完工时间的调度问题,提出了最优的分批策略及批间序的确定方法,并给出一个多项式时间可解的动态规划算法。当工件释放时间和工期同序时,文献[9]证明了极小化最大拖期和拖期工件数问题都是强 NP-难的;文献[10]证明了在链式约束下,即使加工时间都相同,这两个问题仍是强 NP-难的。为了节省能源,提高机器的利用率,与文献[1,8]研究的情况不同,本文每批工件都是一个紧挨着一个进入、离开处理机进行加工,当工件释放时间和加工时间一致时,研究了目标函数为最大完工时间的单机半连续型批处理机调度问题。

1 问题的批释放时间定义及最优解性质

在给出批释放时间的定义之前,先来看下面的例子。

例 1　考虑 7 个工件、批处理机的容量 $C=3$ 的例子,工件的加工时间和释放时间分别为

$p=(1,1,3,3,3,6,6)$,$r=(0,2,3,3,5,6,6)$

若工件 J_1,J_2,J_3,J_4,J_5 放在批 B_1 中加工,J_6,J_7 放在批 B_2 中加工,则批 B_1 的容量即批 B_1 的大小为 5,批 B_2 的容量即批 B_2 的大小为 2。B_1 的基本加工时间为 $p_{(1)}=\max_{J_j\in B_1}\{p_j\}=3$,由于 $C=3$,则 B_1 中工件要每隔 $\frac{p_{(1)}}{C}=1$ 个单位时间一个紧挨着一个地进入处理机加工。若 J_1 在 0 时刻开始加工,则在 1 时刻工件 J_2 要开始加工,但 $r_2=2$,为了保持连续性,批 B_1 中工件 J_1 不能在

$$\max\left\{r_2-\frac{p_{(1)}}{C},r_1\right\}=\max\{2-1,0\}=1$$

时刻之前开始加工,因为在 2 时刻工件 J_2 开始加工,而 $r_3=3$,则工件 J_3 不能在 3 时刻之前开始加工;同上可知,工件 J_4 不能在 4 时刻之前开始加工,工件 J_5 不能在 5 时刻之前开始加工。即为了保持工件的连续性,只需

$$r_5'=r_5=5,\ r_4'=\max\{r_5'-\frac{p_{(1)}}{C},r_4\}=4$$

$$r_3'=\max\{r_4'-\frac{p_{(1)}}{C},r_3\}=3,\ r_2'=\max\{r_3'-\frac{p_{(1)}}{C},r_2\}=2$$

$$r_1'=\max\{r_2'-\frac{p_{(1)}}{C},r_1\}=1$$

定义批 B_1 的批释放时间为 $r_{B_1}=r_1'=1$,由于 $p^{(1)}=p_{(1)}\left(1+\frac{|B_1|-1}{C}\right)=7$,故批 B_1 的完工时间为 8。工件 J_6,J_7 放在批 B_2 中加工,则 B_2 的基本加工时间为 $p_{(2)}=\max_{J_j\in B_2}\{p_j\}=6$,则第二批中的工件要每隔 $\frac{p_{(2)}}{C}=2$ 个单位时间一个紧挨着一个地进入处理机加工。同批 B_1,为了保持连续性

$$r_7'=r_7=6,\ r_6'=\max\left\{r_7'-\frac{p_{(2)}}{C},r_6\right\}=6$$

故批 B_2 的批释放时间

$$r_{B_2}=r_6'=6,\ p^{(2)}=p_{(2)}\left(1+\frac{|B_2|-1}{C}\right)=8$$

则第二批的完工时间为 16。处理过程如图 1。

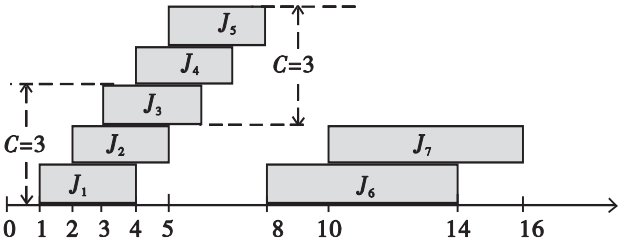

图1 例1的调度图

Fig. 1 Gantt Charts of Example 1

设 $B_k = \{J_{k_1}, J_{k_2}, \cdots, J_{k_{|B_k|}}\}$,注意到批 B_k 中工件的基本加工时间是 $p_{(k)}$,批处理机的容量为 C,因此 B_k 中工件要每隔 $\frac{p_{(k)}}{C}$ 个单位时间一个紧挨着一个地进入处理机加工。若批 B_k 中工件释放时间为 $r = (r_{k_1}, r_{k_2}, \cdots, r_{k_{|B_k|}})$,设 $r'_{k_{|B_k|}} = r_{k_{|B_k|}}$, $r'_{k_{|B_k|-i}} = \max\left\{r'_{k_{|B_k|-i+1}} - \frac{p_{(k)}}{C}, r_{k_{|B_k|-i}}\right\}$, $i = 1, 2, \cdots, |B_k|$,则定义这批工件的批释放时间为 $r_{B_k} = r'_{k_1}$。

性质1 在问题

$$1 \mid c\text{-}batch, C, \mathrm{agr}(r_i, p_i) \mid C_{\max}$$

中,一定存在这样一个最优调度,其中同一批的工件都是按释放时间非减顺序排列。

证明 反证法。假设存在一最优调度,其中第 l 批工件为 $J_{l1}, J_{l2}, \cdots, J_{lk}$ 不是按释放时间非减顺序排列,不妨设工件 J_{li}, J_{lj} $(1 \leqslant i < j \leqslant k)$,有 $r_{li} \geqslant r_{lj}$ 且 J_{li} 在 J_{lj} 前加工。交换这两个工件,即 J_{lj} 在 J_{li} 前加工,由于 $r_{li} \geqslant r_{lj}$,这样交换后,根据批释放时间定义知,第 l 批的批释放时间一定不大于交换前的批释放时间,由于交换工件不改变这批的加工时间,所以交换后第 l 批的完工时间不大于交换前的完工时间,显然交换后目标函数值不增加,故交换后仍然是最优调度,其中同一批的工件都是按工件的释放时间非减顺序排列。 证毕

性质2 在问题 $1 \mid c\text{-}batch, C, \mathrm{agr}(r_i, p_i) \mid C_{\max}$ 中,一定存在这样一个最优调度,批间加工顺序都是按批释放时间非减顺序排列。

证明 反证法。设最优调度 π 中批间加工顺序不按批释放时间非减顺序排列,则存在相邻两批 P, Q,有 $r_{(p)} > r_{(Q)}$ 且批 P 在批 Q 前加工。由批释放时间定义及 $\mathrm{agr}(r_i, p_i)$ 可知,$p_{(p)} > p_{(Q)}$ 其中

$$p_{(P)} = \max_{J_j \in P}\{p_j\},\ p_{(Q)} = \max_{J_j \in Q}\{p_j\}$$

且

$$p^{(P)} = p_{(P)}\left(1 + \frac{|P|-1}{C}\right), p^{(Q)} = p_{(Q)}\left(1 + \frac{|Q|-1}{C}\right)$$

由性质1,不妨设批 P 中第 k 个工件的加工时间大于 $p_{(Q)}$,把批 Q 中所有工件放入批 P 中加工,而原批 P 中后 $|P|-k+1$ 个工件放入批 Q 中,得到新的批 P' 和批 Q',其余工件不变,得到一个新的调度,有

$$p^{(P')} = p_{(Q)}\left(1 + \frac{(k-1) + |Q| - 1}{C}\right) = p_{(Q)}\left(1 + \frac{k + |Q| - 2}{C}\right)$$

$$p^{(Q')} = p_{(P)}\left(1 + \frac{|P| - k + 1 - 1}{C}\right) = p_{(P)}\left(1 + \frac{|P| - k}{C}\right)$$

所以 $p^{(P')} + p^{(Q')} = p_{(Q)}\left(1 + \frac{k + |Q| - 2}{C}\right) + p_{(P)}\left(1 + \frac{|P| - k}{C}\right) = p^{(P)} + p^{(Q)} + \frac{k-1}{C}(p_{(Q)} - p_{(P)})$

而 $p_{(p)} > p_{(Q)}$,则 $p^{(P')} + p^{(Q')} < p^{(P)} + p^{(Q)}$,又因为交换后的调度把释放时间小的工件放在前面加工,所以批 Q' 的完工时间不会大于批 Q 的完工时间,显然交换后目标函数值不增加,即交换后依然是最优调度,其中批间加工顺序都是按批释放时间非减顺序排列。 证毕

性质3 在问题 $1 \mid c\text{-}batch, C, \mathrm{agr}(r_i, p_i) \mid C_{\max}$ 中,一定存在这样一个最优调度,所有工件都是按释放时间非减顺序排列。

证明 由性质2证明过程知,任相邻批按基本加工时间非减排列。设任意两个相邻批 P, Q,都有 $p_{(p)} \leqslant p_{(Q)}$ 且批 P 在批 Q 前加工。对于任意两相邻批 P 和 Q,若批 P 中所有的工件的加工时间都不比批 Q 中的大,由 $\mathrm{agr}(r_i, p_i)$ 可知,此最优调度所有工件都是按释放时间非减顺序排列;否则,不妨设工件 J_i, J_j $(r_i \geqslant r_j, p_i \geqslant p_j)$,且 $J_i \in P$, $J_j \in Q$,交换两个工件的位置,形成新的两批 P' 和 Q',其中工件 $J_i \in Q'$,工件 $J_j \in P'$,交换后,因为 $p_i \leqslant p_{(Q)}$, $p_i \geqslant p_j$,所以 $p_j \leqslant p_{(Q)}$,即 $p_{(p')} \leqslant p_{(P)}$,根据批释放时间定义知,$r_{(p')} \leqslant r_{(P)}$,且把释放时间大的放在批 Q' 中,因此批 Q' 的开始加工时间不大于批 Q 的开始加工时间。又因为 $p_{(P)} \leqslant p_{(Q)}$, $p_i \leqslant p_{(P)}$,所以

$p_i \leqslant p_{(Q)}$,故有 $p_{(Q')} = p_{(Q)}$,即批 Q' 与批 Q 的加工时间相等,故批 Q' 的完工时间不会大于批 Q 的完工时间,显然交换后目标函数值不增加,即交换后的调度仍是一个最优调度,其中所有工件都是按释放时间非减顺序排列。　　证毕

性质 4　在问题 $1|c\text{-}batch, C, \mathrm{agr}(r_i, p_i)|C_{\max}$ 中,一定存在这样一个最优调度,释放时间相同且加工时间也相同的工件在同一批中加工。

证明　用反证法。若结论不成立,则存在一最优调度 π,其中工件 J_i,J_j 有相同的释放时间 r 和加工时间 p,分别在批的加工时间为 $p^{(P)}$、$p^{(Q)}$ 的两个批 P、Q 中加工。由性质 3 知,最优调度 π 中,所有工件都是按释放时间非减顺序排列,因此有批 P 和批 Q 相邻,批 P 在批 Q 前加工,且工件 J_i 是批 P 中最后一个加工的工件,工件 J_j 是批 Q 中第一个加工的工件。现将 J_j 从批 Q 中移入批 P 中,其余工件不动,则批 P 变为批 P',批 Q 变为批 Q',其余批不变,形成新的调度 π',因为 $J_i \in P$,$p_i = p_j = p$,所以

$$p_{(P')} = \max_{J_j \in P'}\{p_j\} = p_{(P)},\ p_{(Q')} = \max_{J_j \in Q'}\{p_j\} = p_{(Q)}$$

而 $|P'| = |P| + 1$,$|Q'| = |Q| - 1$,则

$$p^{(P')} = p_{(P')}\left(1 + \frac{|P'| - 1}{C}\right) = p_{(P)}\left(1 + \frac{|P|}{C}\right)$$

$$p^{(Q')} = p_{(Q)}\left(1 + \frac{|Q| - 2}{C}\right)$$

所以

$$p^{(P')} + p^{(Q')} = p_{(P)}\left(1 + \frac{|P|}{C}\right) + p_{(Q)}\left(1 + \frac{|Q| - 2}{C}\right) = p^{(P)} + p^{(Q)} + \frac{p_{(P)} - p_{(Q)}}{C} \quad (1)$$

由于工件 J_i,J_j 有相同的释放时间 r 且批 P 在批 Q 前加工,所以 J_j 从 Q 中移入 P 后,由批释放时间的定义知,$r_{(P)} = r_{(P')}$。由于批 P' 和批 P 前各批工件不变,故批 P' 和批 P 的开始加工时间相同,因此若批 P 的完工时间为 C_P,则批 P' 的完工时间为 $C_P + \frac{p_{(P)}}{C}$。记批 Q' 的批释放时间为 $r_{(Q')}$,根据批释放时间定义可知 $r_{(Q')} \geqslant r$,且批 Q 的批释放时间是 $r_{(Q)} = \max\left\{r, r_{(Q')} - \frac{p_{(Q)}}{C}\right\}$。若在批 P 和 Q 之间机器有空闲时,即 $r_{(Q')} - \frac{p_{(Q)}}{C} > r$,则批 Q 的批释放时间 $r_{(Q)} = r_{(Q')} - \frac{p_{(Q)}}{C}$,也是批 Q 的开始加工时间,因此 π 中批 P 与 Q 空闲为 $r_{(Q')} - \frac{p_{(Q)}}{C} - C_P$,而 π' 中批 P' 与 Q' 的空闲为 $r_{(Q')} - C_P - \frac{p_{(P)}}{C}$,所以 π' 与 π 空闲差为 $\frac{p_{(Q)} - p_{(P)}}{C}$,即 π' 比 π 的空闲大 $\frac{p_{(Q)} - p_{(P)}}{C}$,由(1)式知,批 Q' 的完工时间等于批 Q 的完工时间,即 $C_{Q'} = C_Q$(如图 2),而其余批均不变。显然交换后目标函数值不增加,即交换后的调度仍是最优调度。若在批 P 和批 Q 之间机器无空闲时,而交换后的新调度 π',有两种情况:1)批 P' 和批 Q' 之间机器仍无空闲,由(1)式知,$C_{Q'} \leqslant C_Q$,而其余批均不变,显然交换后目标函数值不增加,即交换后的调度仍是最优调度;2)若交换后的新调度 π',批 P' 和批 Q' 之间机器产生空闲,则批 Q' 的完工时间 $C_{Q'} = r_{(Q')} + p^{(Q')}$,由于批 Q 的批释放时间是 $\max\left\{r, r_{(Q')} - \frac{p_{(Q)}}{C}\right\}$,则批 Q 的完工时间一定不小于 $\max\left\{r, r_{(Q')} - \frac{p_{(Q)}}{C}\right\} + p^{(Q)}$,而

$$\max\left\{r, r' - \frac{p_{(Q)}}{C}\right\} + p^{(Q)} \geqslant r' - \frac{p_{(Q)}}{C} + p^{(Q)} = r' - \frac{p_{(Q)}}{C} + p_{(Q)}\left(1 + \frac{|Q| - 1}{C}\right) = r_{(Q')} + p^{(Q')} = C_{Q'}$$

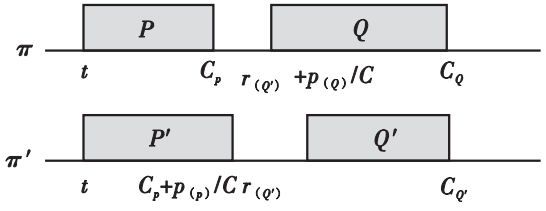

图 2　π 和 π' 的调度图

Fig. 2　Gantt Charts of π and π'

故交换后批 Q' 的完工时间不大于批 Q 的完工时间(如图 3),而其余批均不变,显然交换后目标函数值不增加,故交换后的调度仍是最优调度,即一定存在这样一个最优调度,释放时间相同且加工时间也相同的工件在同一批中加工。　　证毕

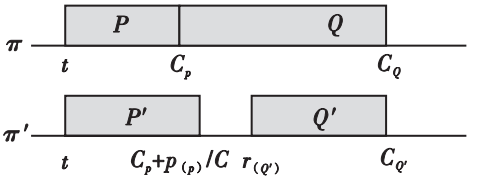

图3 批 P 与 Q 重组的调度图

Fig. 3 Gant Charts of P and Q recombined

根据上述性质直接可以得到下面的推论。

推论1 在问题 $1|c-batch,C,\mathrm{agr}(r_i,p_i)|C_{\max}$ 中,一定存在这样的最优调度,其同一批的工件是释放时间相邻的,即若 $r_{j_1}\leqslant r_{j_2}\leqslant\cdots\leqslant r_{j_k}$,其中 $k\leqslant n$,而 $J_{j_1},J_{j_2},\cdots,J_{j_{i-1}},J_{j_{i+1}},\cdots,J_{j_k}$ 在同一批中,则 J_{j_i} 必在此批中加工。

2 动态规划算法

由问题 $1|c-batch,C,\mathrm{agr}(r_i,p_i)|C_{\max}$ 的性质可知,这个问题的所有最优调度都可以转化为工件按释放时间非减顺序排列,即 $J_1,J_2,\cdots,J_n$ 使得 $r_1\leqslant r_2\leqslant\cdots\leqslant r_n$。工件按释放时间非减顺序排列后,要对工件进行分批。下述动态规划算法实际上就是求从第一个工件 J_1 到任一个工件的最优分批。其思想是,当工件 $J_1,\cdots,J_{k-1}$ 已进行了最优分批时,现增加工件 J_k 后,工件 J_k 或者与前一批中若干个工件形成一批或者单独形成一批进行加工,从中比较后取最小值。

事实上,如果 $B_1,B_2,\cdots,B_p$ 是工件从 J_1 到 J_n 的最优分批,如果批 $B_j(j=1,2,\cdots,p)$ 的最后一个工件为 $J_k(k=1,2,\cdots,n)$,那么工件从 J_1 到 J_k 按 $B_1,B_2,\cdots,B_j$ 分批最优。如果这个结论不成立,设 $\overline{B}_1,\overline{B}_2,\cdots,\overline{B}_l$ 是工件从 J_1 到 J_k 的最优分批,令工件 J_1 到 J_k 按 $\overline{B}_1,\overline{B}_2,\cdots,\overline{B}_l$ 分批,工件 J_{k+1} 到 J_n 按 $B_{j+1},\cdots,B_p$ 分批,则这种分批方法对应的目标函数值比前一种分批方法对应目标函数值要小,但这与 $B_1,B_2,\cdots,B_p$ 是工件从 J_1 到 J_n 对应的最优分批矛盾。

在下述动态规划算法中,$f(k)$ 表示工件从 J_1 到 J_k 对应的目标函数值,B_k' 是按 $f(k)$ 分批的最后一批,$\mathrm{agr}\min\{\varphi(i)\}$ 表示使 $\varphi(i)$ 取最小值对应的 i 值。

动态规划算法1 步骤1 把工件按 $r_1\leqslant r_2\leqslant\cdots\leqslant r_n$ 序编号;

步骤2 令 $f(0)=0,B_0'=\varnothing,s_0=0,k=1$;

步骤3 按递归方程

$$f(k)=\min_{1\leqslant i\leqslant s_{k-1}+1}\left\{\max\{f(k-i),r'_{k-i+1}\}+p_k\left(1+\frac{i-1}{C}\right)\right\}$$

计算 $f(k),k=1,2,\cdots,n$;其中 $r_k'=r_k$,当 $2\leqslant i\leqslant s_{k-1}+1$ 时,$r'_{k-i+1}=\max\left\{r'_{k-i+2}-\frac{p_k}{C},r_{k-i+1}\right\}$,令

$$s_k=\operatorname*{agr\,min}_{1\leqslant i\leqslant s_{k-1}+1}\left\{\max\{f(k-i),r'_{k-i+1}\}+p_k\left(1+\frac{i-1}{C}\right)\right\}$$

$$B_k'=\{J_{k-s_k+1},\cdots,J_k\}$$

步骤4 依次安排各批次序。

求出 $f(n)$ 后,利用反向追踪就得到了所有工件的一个最优分批,$f(n)$ 就是最优值。

下面证明算法的最优性。

定理 工件释放时间和加工时间一致时,对于目标函数为最大完工时间的调度问题,动态规划算法1可得到其调度的最优解。

证明 由推论3知,只要证明对 $r_1\leqslant r_2\leqslant\cdots\leqslant r_n$ 序排列后的工件进行分批即可。下面用数学归纳法来证明这个结论。

当 $k=1$ 时,显然成立。

当 $k=2$ 时,只能有两种分批情况:J_1,J_2 分在一批或各自组成一批处理,比较它们的大小就可得到问题的解,这在算法中已经体现。结论成立。

假设 $k-1$ 时结论成立,即利用动态规划算法1能得到问题的最优分批。不失一般性,设工件按 $r_1\leqslant r_2\leqslant\cdots\leqslant r_k$ 排序后的顺序为 $J_1,J_2,\cdots,J_k$;最优分批为 $\pi:B_1,B_2,\cdots,B_l$;且 $B_l=\{J_{s+1},J_{s+2},\cdots,J_{k-1}\}$。下面证明结论对 k 个工件也成立。

设按动态规划算法1可得到 k 个工件的一个最优分批的最后一批为 B_q,则 $B_q=\{J_{s'+1},J_{s'+2},\cdots,J_k\}$ 且 $B_q\subseteq B_l\cup\{J_k\}$,也就是 $s'\geqslant s$。下面用反证法证明这一结论。

假设上述结论不成立,即 $s'<s$,由假设各工件按 $r_1\leqslant r_2\leqslant\cdots\leqslant r_n$ 排列,所以

$$r_{s'}\leqslant r_{s'+1}\leqslant r_s\leqslant r_{s+1} \tag{2}$$

此时设 $f'(k)$ 表示工件从 J_1 到 J_k 最后一批为 B_q 的目标函数值,其中 $B_q=\{J_{s'+1},J_{s'+2},\cdots,J_k\}$,则有

$$f'(k)=\max\{f(s'),r'_{s'+1}\}+p_k\left(1+\frac{k-s'-1}{C}\right)=$$

$$\max\{f(s'),r'_{s'+1}\}+p_{k-1}\left(1+\frac{k-s'-2}{C}\right)+$$

$$(p_k-p_{k-1})\left(1+\frac{k-s'-2}{C}\right)+\frac{p_k}{C} \tag{3}$$

下面分两种情况讨论。

1)当 $r_k-\frac{p_k}{C}\leqslant r_{k-1}$ 即 $r_k\leqslant r_{k-1}+\frac{p_k}{C}$ 时,表示

$r_k \leqslant C_{k-1}$ (C_{k-1} 是工件 $k-1$ 开始加工时间与机器上实际加工时间之和),即机器加工完工件 J_{k-1} 后可以直接加工工件 J_k ,则由批释放时间定义可知,前 k 个工件最后一批为 $\{J_{s+1},J_{s+2},\cdots,J_k\}$ 的释放时间 r'_{s+1} 不大于前 $k-1$ 个工件最后一批为 $\{J_{s+1},J_{s+2},\cdots,J_{k-1}\}$ 的释放时间 $\bar{r}'_{s+1}$,即 $r'_{s+1} \leqslant \bar{r}'_{s+1}$,故(3)式的右边前两项是前 $k-1$ 个工件的目标函数值,而

$$f(k-1) = \max\{f(s),\bar{r}'_{s+1}\} + p_{k-1}\left(1+\frac{k-s-2}{C}\right) \tag{4}$$

是前 $k-1$ 个工件的目标函数的最优值,又因为 $p_{k-1} \leqslant p_k$,所以有

$$\max\{f(s'),r'_{s'+1}\} + p_k\left(1+\frac{k-s'-2}{C}\right) \geqslant f(k-1) \tag{5}$$

由(3)、(5)式得

$$f'(k) \geqslant f(k-1) + (p_k - p_{k-1}) \cdot \left(1+\frac{k-s'-2}{C}\right) + \frac{p_k}{C} \tag{6}$$

因为 $s' < s$,把(4)式代入(6)式得

$$f'(k) \geqslant \max\{f(s),\bar{r}'_{s+1}\} + p_{k-1}\left(1+\frac{k-s-2}{C}\right) + (p_k - p_{k-1})\left(1+\frac{k-s'-2}{C}\right) + \frac{p_k}{C} >$$

$$\max\{f(s),\bar{r}'_{s+1}\} + p_{k-1}\left(1+\frac{k-s-2}{C}\right) + (p_k - p_{k-1})\left(1+\frac{k-s-2}{C}\right) + \frac{p_k}{C} = \max\{f(s),\bar{r}'_{s+1}\} + p_k\left(1+\frac{k-s-1}{C}\right) \geqslant$$

$$\max\{f(s),r'_{s+1}\} + p_k\left(1+\frac{k-s-1}{C}\right) \tag{7}$$

(7)式中不等式的右端是将 J_k 直接并入 B_l 内处理,而其余工件按 $B_1,B_2,\cdots,B_{l-1}$ 分批对应的目标函数值。从(7)式可以看出假设不成立,即 $s' \geqslant s$。

2)当 $r_k - \frac{p_k}{C} > r_{k-1}$ 时,把前 $k-1$ 个工件中 $\{J_{s+1},J_{s+2},\cdots,J_{k-1}\}$ 作为最后一批时,若增加一个工件 J_k 后使机器在批 B_l 前依然无空闲或者机器空闲的时间不增加,则与情况 1)相同,即证。

若增加一个工件 J_k ,最后一批为 $\{J_{s+1},J_{s+2},\cdots,J_k\}$ 使机器产生空闲或者机器空闲的时间增大时,目标函数值为

$$\max\{f(s),r'_{s+1}\} + p_k\left(1+\frac{k-s-1}{C}\right) = r'_{s+1} + p_k\left(1+\frac{k-s-1}{C}\right)$$

最后一批工件为 $\{J_{s'+1},J_{s'+2},\cdots,J_k\}$ 时,目标函数值

$$f'(k) = \max\{f(s'),r'_{s'+1}\} + p_k\left(1+\frac{k-s'-1}{C}\right) \geqslant r'_{s'+1} + p_k\left(1+\frac{k-s'-1}{C}\right) = r'_{s'+1} + p_k\frac{s-s'}{C} + p_k\left(1+\frac{k-s-1}{C}\right) \tag{8}$$

由动态规划算法 1 中 $r'_{k-i+1} = \max\{r'_{k-i+2} - \frac{p_k}{C}, r_{k-i+1}\}$,可知

$$r'_{s'+1} \geqslant r'_{s+1} - p_k\frac{s-s'}{C} \tag{9}$$

把(9)式代入(8)式中得

$$f'(k) \geqslant r'_{s+1} + p_k\left(1+\frac{k-s-1}{C}\right) \tag{10}$$

(10)式中不等式的右端是将 J_k 直接并入 B_l 内处理,而其余工件按 $B_1,B_2,\cdots,B_{l-1}$ 分批对应的目标函数值。从(10)式可以看出假设不成立,即 $s' \geqslant s$。

因此增加 J_k 工件后,最优调度的最后一批只能有下列 3 种情况:

ⅰ) J_k 被分在 B_l 中加工;

ⅱ) J_k 和 B_L 中的部分工件作为一批加工,即在批 B_l 中某个工件 J_r 处断开,($J_{r+1},\cdots,J_k$)为 k 个工件 J_1到 J_k 最优分批的最后一批;

ⅲ) J_k 单独形成一批加工。而动态规划算法 1 是对这 3 种情况比较后取最小值。所以结论成立。证毕

由于算法 1 中动态规划过程每次循环需要 $O(k)$ 次加减法、$O(k)$ 次乘除法和 $O(k)$ 次比较,故共需 $O(n^2)$ 次加减法、$O(n^2)$ 次乘除法和 $O(n^2)$ 次比较,那么动态规划过程的计算复杂性是 $O(n^2)$,而对工件进行排序的计算复杂性是 $O(n\log n)$,所以此算法的计算复杂性为 $O(n^2)$ 。而且本算法在增加一个工件后进行分批时,只从未增加这个工件的分批中最后一批工件进行讨论,这样计算量减少了许多,尤其当分的批数越多时,算法的优势越明显。

3 数值例子

例 2　考虑 9 工件, $C=4$ 的 1 个例子,其中工

件的加工时间和释放时间分别为

$$p = (1,1,1,1,2,4,4,6,10)$$

$$r = (0,1,1,2,2,3,5,8,10)$$

用上述动态规划算法计算,过程如下。

由初始条件得:$f(0) = 0, s_0 = 0, B_0 = \varnothing, k = 1$。计算$f(1)$:因为$p_1 = 1, r_1 = 0$,则

$r_1' = r_1 = 0$, $f(1) = \min\{f(0), r_1'\} + p_1 = 0 + 1 = 1$

$$s_1 = 0, r_{B_1'} = 0, B_1' = \{J_1\}$$

当$k = 2$时,$r_2' = r_2 = 1, r_1' = \max\left\{r_2 - \frac{p_2}{C}, r_1\right\} = \max\left\{1 - \frac{1}{4}, 0\right\} = \frac{3}{4}$,所以

$$f(2) = \min\left\{\max\{f(0), r_1'\} + p_1\left(1 + \frac{1}{C}\right), \max\{f(1), r_2'\} + p_1\right\} = \min\{2,2\} = 2$$

$$s_2 = 0, r_{B_2'} = \frac{3}{4}, B_2' = \{J_1, J_2\}$$ 或

$$s_2 = 1, r_{B_2'} = 1, B_2' = \{J_2\}$$

当$k = 3$时,

$$r_3' = r_3 = 1, r_2' = \max\left\{r_3' - \frac{p_3}{C}, r_2\right\} = \max\left\{1 - \frac{1}{4}, 1\right\} = 1, r_1' = \max\left\{r_2' - \frac{p_3}{C}, r_1\right\} = \max\left\{1 - \frac{1}{4}, 0\right\} = \frac{3}{4}$$

$$f(3) = \min\left\{\max\{f(0), r_1'\} + p_3\left(1 + \frac{2}{C}\right), \max\{f(1), r_2'\} + p_3\left(1 + \frac{1}{C}\right), \max\{f(2), r_3'\} + p_3\right\} = \min\left\{\frac{9}{4}, \frac{9}{4}, 3\right\} = \frac{9}{4}, s_3 = 0, r_{B_3'} = \frac{3}{4}, B_3' = \{J_1, J_2, J_3\}$$ 或 $s_3 = 1, r_{B_3'} = 1, B_3' = \{J_2, J_3\}$

重复上述的过程可得

$$f(4) = \min\left\{3,3,\frac{13}{4},\frac{13}{4}\right\} = 3, s_4 = 0, r_{B_4'} = \frac{5}{4}, B_4' = \{J_1, J_2, J_3, J_4\}$$ 或 $s_4 = 1, r_{B_4'} = \frac{3}{2}, B_4' = \{J_2, J_3, J_4\}$

$$f(5) = \min\left\{\frac{9}{2}, \frac{9}{2}, 5, \frac{19}{4}, 5\right\} = \frac{9}{2}, s_5 = 0, r_{B'_5} = \frac{1}{2}, B_5' = \{J_1, J_2, J_3, J_4, J_5\}$$ 或 $s_5 = 1, r_{B_5'} = 1, B_5' = \{J_2, J_3, J_4, J_5\}$

$$f(6) = \min\left\{9,9,9,\frac{33}{4},8,\frac{17}{2}\right\} = 8$$

$$s_6 = 4, r_{B_6'} = 2, B_6' = \{J_5, J_6\}$$

$$f(7) = \min\left\{9,\frac{19}{2},12\right\} = 9, s_7 = 4$$

$$r_{B_7'} = 3, B_7' = \{J_5, J_6, J_7\}$$

$$f(8) = \min\left\{14,14,\frac{31}{2},15\right\} = 14$$

$$s_8 = 4, r_{B_8'} = \frac{7}{2}, B_8' = \{J_5, J_6, J_7, J_8\}$$

$$f(9) = \min\left\{23,22,23,\frac{43}{2},24\right\} = \frac{43}{2}, s_9 = 5, r_{B_9'} = 8, B_9' = \{J_8, J_9\}$$

最优值为$\frac{43}{2}$,通过反向追踪得到所有工件的一个最优分批为$\{J_1, J_2, J_3, J_4\}$,$\{J_5, J_6, J_7\}$,$\{J_8, J_9\}$。最优调度如图4所示。

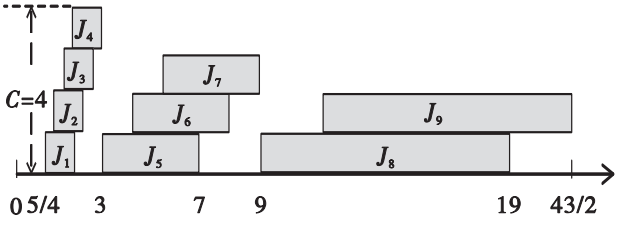

图4 例2的调度图

Fig. 4 Gantt Charts of Example 2

4 结论

在实际问题中,加热炉加热只是中间生产环节,工件带有释放时间的问题更为普遍,因此研究这类问题具有较强的实际应用背景。这类问题的讨论,涉及工件如何分批、各批间的顺序和批内的顺序等问题。本文对于释放时间和加工时间一致的极小化最大完工时间问题,给出复杂性为$O(n^2)$的多项式时间算法。在此问题中还有机器允许有空位、工件分簇等情况,其他目标函数值,如极小化完工时间、拖期等问题,因此对于这一模型研究值得进一步探讨。

参考文献:

[1] 赵玉芳,唐立新. 极小化最大完工时间的单机连续型批调度问题[J]. 自动化学报,2006,32(5):730-737.

[2] 唐国春. 现代排序论[M]. 上海:上海科学普及出版社, 2003.

[3] Lee C Y, Uzsoy R, Martin-Vega L A. Efficient algorithms for scheduling semiconductor burn-in operations[J]. Operations Research, 1992, 40: 764-775.

[4] Lee C Y, Uzsoy R. Minimizing makespan on a single batch processing machine with dynamic job arrivals[J]. International Journal of Production Research, 1999, 37: 219-236.

[5] Liu Z H, Yu W C. Scheduling one batch processor subject to job release dates[J]. Discrete Applied Mathematics, 2000, 105: 129-136.

[6] 刘朝晖, 俞文. 无容量限制的批处理机时间表问题[J]. 华东理工大学学报, 2001, 27(4): 431-433.

[7] Ng C T, Cheng T C E, Yuan J J, et al. On the single machine serial batching scheduleing problem to minimize total completeon time with precedence constraints, releasee dates and identicall processing times[J]. Operations Research Letters, 2003, 31: 323-326.

[8] 赵玉芳, 唐立新. 极小化总完工时间的单机连续型批调度问题[J]. 电子学报, 2008, 36(2): 367-370.

[9] 赵玉芳, 唐立新. 释放时间和工期同序的单机连续型批调度问题[J]. 自动化学报, 2008, 34(8): 957-963.

[10] 赵玉芳, 唐立新. 链式约束下释放时间和工期同序的单机连续型批调度问题[J]. 控制与决策, 2008, 23(6): 677-680.

Operations Research and Cybernetics

Semicontinuous Batch Processor Scheduling with Release Time

WANG Song-li, ZHAO Yu-fang, CUI Miao-miao

(School of Mathematics and System Science, Shenyang Normal University, Shenyang 110034, China)

Abstract: We consider the problem of semicontinuous batch scheduling arisen in the heating-process of blooms in the iron and steel industry. Jobs are processed in batch, the processing time of jobs in the same batch is the longest processing time of jobs in the batch and the jobs enter or leave the machine one after another in periods. The capacity of the machine is C, which can handle up to C jobs simultaneously. The capacity of a batch is the size of batch; the processing time of a batch is related to its size, the longest processing time of jobs in the batch and the capacity of the machine. In this paper, we assume that the job release time and processing time are agreeable. For the problem to minimize makespan, we study the characterizations of the optimal batching and then turn into nondecreasing order of the release time, process in batches and obtain the minimize makespan. On that basis, we present a dynamic programming algorithm with complexity of $O(n^2)$ and prove properties of the optimal solution, and give a numerical example to explain the process of algorithm.

Key words: heating-furnace scheduling; semicontinuous batch; computational complexity; dynamic programming algorithm

(责任编辑 黄 颖)