图形处理中一类 Flow-shop 问题的改进算法

蒋义伟 [1] 魏 麒 [2]

摘 要 考虑图形处理中的一类两台处理器上的 Flow-shop 调度问题, 目标是极小化最早完工时间. 每个任务包含两道工序, 第一道工序可以在两台处理器中的任何一台上处理, 而第二道则只能在第二台处理器上处理, 且必须在第一道工序完工之后才能进行. 对该问题, 设计了一个改进的多项式时间近似算法, 在绝对性能方面, 该算法的最坏情况界为 3/2; 而从实例计算的平均效果方面, 该算法所得的结果比原有的贪婪算法所得的结果要好 20 % 左右.

关键词 调度, 近似算法, 最早完成时间, 流水作业

DOI 10.3724/SP.J.1004.2011.01381

An Improved Algorithm for a Hybrid Flow-shop Problem in Graphics Processing

JIANG Yi-Wei[1] WEI Qi[2]

Abstract This paper considers a variant of scheduling problem of minimizing makespan in a two-machine flow-shop. In this variant, each job has two tasks. The first task can be processed on either machine while the second task must be processed on the second machine and cannot be processed unless the first task has been processed. The second task is allowed to be processed at any time after completing the first task. We present an improved polynomial time approximation algorithm with worst-case ratio of 3/2, which is 20 % better than the greedy-like algorithm in the average case.

Key words Scheduling, approximation algorithm, makespan, flow-shop

本文考虑一类特殊 Flow-shop 调度模型, 该模型主要应用于计算机图形处理中[1], 图形程序处理包括前期的数据处理和后期的图形处理, 后者必须在前者处理完毕后才能处理. 数据处理既可以通过 CPU 也可以通过 GPU 进行处理, 而图形处理则只能通过 GPU. 目标是通过合理安排使得所有图形都处理完的时间尽可能的早. 因此我们将 CPU 和 GPU 看作两台 Flow-shop 处理器, 将数据处理和图形处理看成两道不同的工序, 就可转化为我们所考虑的调度问题. 具体描述如下: 有 n 个任务 J_1, J_2, $\cdots$, J_n 必须在两台处理器 M_1 和 M_2 上进行处理. 每个任务 J_i 包含两道工序 A_i 和 B_i, $i = 1, 2, \cdots, n$. 工序 A_i 可以在处理器 M_1 或 M_2 上处理, 处理时间为 a_i, 而工序 B_i 只能在 M_2 上处理, 时间为 b_i, 并且只能在工序 A_i 完成后才能进行处理. 假设所有的任务在零时刻就已知并且工序在处理过程中不允许中断. 对一个给定的调度计划, 记 $S_{A_i}(S_{B_i})$ 和 $C_{A_i}(C_{B_i})$ 为工序 $A_i(B_i)$ 的开工时间和完工时间. 目标是极小化所有任务的最大完工时间, 即 $C_{\max} = \max\{C_{B_i}\}$.

从上述描述可以看出本文考虑的问题与经典的两台处理器 Flow-shop 调度问题不同, 经典的问题每个任务的加工方式都是相同的, 即都是第一道工序在第一台机, 第二道工序在第二台机上处理. 与这类问题较为相关的模型还有 Kouvelis 等[2] 提出的一类混合 Flow-shop 问题 (Hybrid flow-shop, HFS), 该问题假设任务的两道工序都可以在同一台机器上进行处理或者第一道工序在第一台机器, 第二道工序在第二台机器上处理, 不难看出任务的处理方式有三种. 文中证明了该问题是 NP 难的, 并给出了一个基于动态规划的最优算法和一个伪多项式时间近似算法. Vairaktarakis 等[3] 考虑了另外的情况, 第二道工序必须在第一道工序完工后才能加工, 第一道工序可以单独在第一台机器上处理, 也可以用较少的时间在两台机器上同时进行并行处理, 第二道工序可以单独在第二台机器上处理, 也可以用较少的时间在两台机器上同时进行处理, 他们给出了一个动态规划算法和一个最坏情况界为 1.618 的多项式时间近似算法. 文献 [4] 则考虑了一类有回流的混合 Flow-shop 问题, 分别考虑了在第二阶段为单机和两台同型机的情形, 对于前者证明了多项式时间可解并给出了算法, 对于后者证明了该情况是 NP 难的, 给出了动态规划和启发式算法. 此外还有文献研究带机器不确定维护时间的 Flow-shop 问题[5]. 关于 Flow-shop 调度问题, 还有很多文献通过

收稿日期 2011-01-21 录用日期 2011-06-13
Manuscript received January 21, 2011; accepted June 13, 2011
国家自然科学基金 (11001242, 11071220), 浙江省自然科学基金 (Y6090175, Y6090554) 资助
Supported by National Natural Science Foundation of China (11001242, 11071220) and Zhejiang Province Natural Science Foundation of China (Y6090175, Y6090554)
1. 浙江理工大学理学院 杭州 310018 2. 浙江大学宁波理工学院 宁波 315100
1. Faculty of Science, Zhejiang Sci-Tech University, Hangzhou 310018 2. Ningbo Institute of Technology, Zhejiang University, Ningbo 315100

遗传算法等优化方法来求解[6−10], 文献 [11] 则考虑了任务具有三道工序的两台机混合流水作业问题, 分别给出了近似算法和多项式时间近似方案 (Polynomial time approximation scheme, PTAS).

本文所考虑的问题最早由 Wei 等[1] 提出, 记为半混合流水作业 (Semi-hybrid flow-shop, SHFS). 他们证明了该问题是 NP 难的, 文献 [12] 证明该问题贪婪算法的最坏情况界为 5/3, 并给出了一个最坏情况界为 8/5 的改进算法. 本文接下来将基于文献 [11] 的思想给出一个更好的多项式时间近似算法, 该算法提出四种调度方案并从中选择结果最好的作为输出. 在近似算法的绝对性能方面, 证明了最坏情况界为 3/2; 而在平均效果方面, 通过实例计算发现, 该算法所得的结果要比原有的贪婪算法的结果要好 20 % 左右.

1 问题的初步分析

由问题的描述不难看出, 每个任务的处理方式有两种选择: 一种是将两道工序 A_i 和 B_i 都放在机器 M_2 上处理, 处理时间分别为 a_i 和 b_i, 记这种方式为模式 1; 另一种称为模式 2, 即工序 A_i 先在 M_1 上处理, 时间为 a_i, 然后 B_i 在 M_2 上处理, 时间为 b_i. 因此, 对任意一个调度, 根据任务处理模式的不同, 可以将任务集 V 分成两个不交子集 V_1 和 V_2, 其中 V_i 中的任务的处理模式为模式 i, $i = 1, 2$.

注意到集合 V_1 中的任务的两道工序都是在 M_2 上处理, 因此对 $J_i \in V_1$, 可以将该任务在两台机器上的处理时间分别记为 $p_i^1 = 0$ 和 $p_i^2 = a_i + b_i$, 而对 $J_i \in V_2$, 在两台机器上的处理时间记作 $p_i^1 = a_i$ 和 $p_i^2 = b_i$. 不难看出, 在集合划分给定的情况下, 通过上述转化, 问题 SHFS 可以看做经典的两台机 Flow-shop 问题. 因此可以得到如下结论:

引理 1. 若任务集 V 已划分为 V_1 和 V_2 两个子集, 则 SHFS 可由 Johnson 规则[13] 在 $\mathrm{O}(n \log n)$ 时间内给出最优调度.

由此不难得到如下结论:

引理 2. 存在一个 SHFS 的最优调度 (图 1) 满足以下性质:

1) 所有 V_1 中的任务都在 V_2 中的任务之前完工;

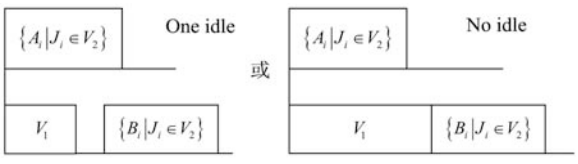

图 1 满足引理 2 的最优调度

Fig. 1 Optimal scheduling satisfying Lemma 2

2) V_2 中的任务按照 Johnson 规则进行处理;

3) 机器 M_1 不存在空闲时间;

4) 机器 M_2 至多只有一个空闲时间段, 若存在则必在 V_1 的最后一个任务和 V_2 中的第一个任务之间.

若 V_1 中任务的两道工序之间允许插入其他任务的工序, 记这样的 SHFS 问题为 ReSHFS. 下面的引理说明 ReSHFS 并不会降低原问题的最优目标值.

引理 3. ReSHFS 的最优目标值不会小于原问题 SHFS 的最优目标值.

证明. 设 S 是 ReSHFS 关于给定 V_1 和 V_2 的一个最优调度. 对任意的 $J_i \in V_1$, 若 A_i 和 B_i 之间存在其他工序, 则将工序 B_i 移到 A_i 的后面, 而之间的工序都按序往后移. 显然经过这样的调整后, 该调度仍然可行, 并且最优目标值不会改变, 因此调整后的调度还是 ReSHFS 的最优调度. 进一步, 调整后的调度是 SHFS 的一个可行调度. 由此可得 ReSHFS 的最优目标值不会小于 SHFS 的最优目标值. □

由上面结论可以看出, SHFS 问题可以找到最优调度, 使得在 V_1 中任务的两道工序之间不存在其他任务的工序, 因此, 可以说 ReSHFS 调度是 SHFS 的一个可行调度.

分别记 $C_{\max}(S(H))$ 和 $C_{\max}(S^*)$ 为近似算法 H 的目标值和最优目标值. 下面的引理给出了最优目标值的下界.

引理 4. 问题 SHFS 的最优目标值

$$C_{\max}(S^*) \geq \max\left\{\max_{1\leq i\leq n}(a_i + b_i), \frac{1}{2}\sum_{i=1}^{n}(a_i + b_i)\right\}$$

证明. 由于每个任务的处理时间为 $a_i + b_i$, 则第一个式子显然成立. 又因为是两台机器加工, 因此最优目标值至少为总处理时间的平均值, 即 $C_{\max}(S^*) \geq \frac{1}{2}\sum_{i=1}^{n}(a_i + b_i)$. □

2 近似算法

本节将给出问题 SHFS 的一个多项式时间近似算法, 并证明其最坏情况界为 3/2. 进一步, 通过实例计算可以发现, 该算法不仅在绝对性能比上进行了改进, 而且在平均情况下, 目标值比原有的算法要小近 20 %.

在给出算法之前, 我们首先引入经典的两台机平行机排序问题, 即 $P2\|C_{\max}$. 将问题 SHFS 的 n 个任务看成 $P2\|C_{\max}$ 的 n 个任务, 任务 J_j 的处理时间为第一道工序的时间 a_j, 对这些任务应用 LS 算法[14] 进行调度, 记 J_r 是决定目标值的任务. 把与 J_r 在同一台机器上处理的任务 (J_r 除外) 的集合记为 V', 另一台机机器上的任务记为 V''.

由引理 1 可知, 任务集一旦已经划分, 只需用 Johnson 规则便可得到给定划分的最优调度, 因此解决 SHFS 问题的关键在于集合的划分, 即任务的处理模式的划分.

接下来, 我们给出下面四种划分 PS_i, $i = 1, 2, 3, 4$.

1) PS_1: $V_1 = V$, $V_2 = \varnothing$, 即所有任务都按模式 1 处理;

2) PS_2: $V_1 = \varnothing$, $V_2 = V$, 即所有任务都按模式 2 处理;

3) PS_3: $V_1 = V^{''} \cup \{J_r\}$, $V_2 = V^{'}$, 即 $V^{''}$ 中的任务和 J_r 按模式 1 处理, $V^{'}$ 中的任务按模式 2 处理;

4) PS_4: $V_1 = V^{''}$, $V_2 = V^{'} \cup \{J_r\}$, 即 $V^{''}$ 中的任务按模式 1 处理, $V^{'}$ 中的任务和 J_r 按模式 2 处理.

根据上述划分, 给出多项式时间近似算法 H.

算法 H.

步骤 1. 将任务进行划分得到上述四种情况 PS_i.

步骤 2. 对每一种划分 PS_i, $i = 1, 2, 3, 4$.

步骤 2.1. 若 $J_i \in V_1$, 记第一道工序处理时间为 $p_i^1 = 0$, 第二道工序处理时间为 $p_i^2 = a_i + b_i$;

步骤 2.2. 若 $J_i \in V_2$, 记第一道工序处理时间为 $p_i^1 = a_i$, 第二道工序处理时间为 $p_i^2 = b_i$;

步骤 2.3. 按照 Johnson 规则进行两台机经典的 Flow-shop 调度, 由此可得到相应的 SHFS 的调度 $S(PS_i)$.

步骤 3. 输出上面四种调度的最小目标值作为结果, 即 $C_{\max}(S(H)) = \min_{1 \le i \le 4} C_{\max}(S(PS_i))$.

2.1　最坏情况界分析

定理 1. $C_{\max}(S(H))/C_{\max}(S^*) \le 3/2$, 且界是紧的.

证明. 设 S^* 是问题 SHFS 的一个最优调度, 其中按照模式 1 和模式 2 处理的任务集分别记为 V_1^* 和 V_2^*. 下面根据最优目标值的大小分两种情况进行讨论.

情况 1. $\sum_{i=1}^{n} a_i \le C_{\max}(S^*)$. 这里我们考虑前两种调度 $S(PS_1)$ 和 $S(PS_2)$.

考虑第一种调度 $S(PS_1)$. 先根据 S^* 构造 ReSHFS 的一个调度 S_1: 所有 $J_i \in V_2^*$ 的工序 A_i (即原来在 M_1 上的那些工序) 转到 M_2 上从零时刻开始处理, 而原来 S^* 中在 M_2 处理的依次向后移 (如图 2). 不难看出调度 S_1 中任务的处理模式与 $S(PS_1)$ 是相同的, 即所有任务都是按照模式 2 进行处理. 因 S_1 是 $S(PS_1)$ 的一个 ReSHFS 调度, 根据引理 3 可知, $S(PS_1)$ 的目标值不会大于 S_1 的目标值, 即

$$C_{\max}(S(PS_1)) \le C_{\max}(S_1) = C_{\max}(S^*) + \sum_{J_i \in V_2^*} a_i \tag{1}$$

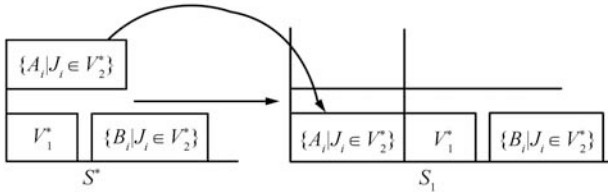

图 2　由 S^* 得到的一个调度 S_1

Fig. 2　Schedule S_1 from S^*

接下来考虑第二种调度 $S(PS_2)$. 同样先得到 S^* 的一个 ReSHFS 调度 S_2: 将所有 $J_i \in V_1^*$ 的工序 A_i 放在 M_1 上从零时刻开始处理, 而原来在 M_1 的工序则在其后面接着处理, 即所有 $J_i \in V_2^*$ 的 A_i 按在 S^* 中的顺序从时刻 $\sum_{J_i \in V_1^*} a_i$ 开始处理, 所有 $J_i \in V$ 的工序 B_i 按照 S^* 的顺序依次在 M_2 上从 $\sum_{J_i \in V_1^*} a_i$ 时刻开始处理 (如图 3).

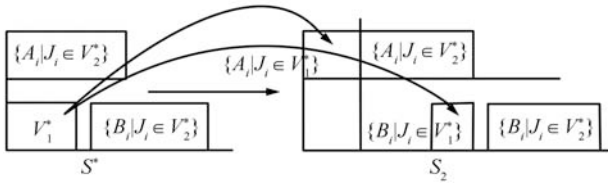

图 3　由 S^* 得到的一个调度 S_2

Fig. 3　Schedule S_2 from S^*

显然, S_2 是 $S(PS_2)$ 的一个 ReSHFS 调度, 因此由引理 3 可得, $S(PS_2)$ 的目标值不大于 S_2 的目标值, 即

$$C_{\max}(S(PS_2)) \le C_{\max}(S_2) = C_{\max}(S^*) + \sum_{J_i \in V_1^*} a_i \tag{2}$$

由式 (1) 和式 (2), 可得

$$\begin{aligned} &C_{\max}(S(PS_1)) + C_{\max}(S(PS_2)) \le \\ &\quad 2C_{\max}(S^*) + \sum_{J_i \in V_2^*} a_i + \sum_{J_i \in V_1^*} a_i = \\ &\quad 2C_{\max}(S^*) + \sum_{i=1}^{n} a_i \end{aligned}$$

由情况 1 的条件可知, $\sum_{i=1}^{n} a_i \le C_{\max}(S^*)$, 故

$$C_{\max}(S(PS_1)) + C_{\max}(S(PS_2)) \le 3C_{\max}(S^*)$$

因此

$C_{\max}(S(H)) \leq$

$$\min\{C_{\max}(S(PS_1)), C_{\max}(S(PS_2))\} \leq \frac{3}{2}C_{\max}(S^*)$$

情况 2. $\sum_{i=1}^{n} a_i > C_{\max}(S^*)$. 这里我们考虑后两种调度 $S(PS_3)$ 和 $S(PS_4)$.

根据 LS 算法的规则不难得到

$$\sum_{J_i \in V'} a_i \leq \sum_{J_i \in V''} a_i \leq \sum_{J_i \in V'} a_i + a_r \tag{3}$$

先考虑 $S(PS_3)$. 构造一个 $S(PS_3)$ 的 ReSHFS 调度 S_3: 在 M_1 上, 从时刻 a_r 按任意顺序处理所有 $J_i \in V^{'}$ 的工序 A_i; 在 M_2 上, 先在零时刻处理工序 A_r, 接着处理 $J_i \in V^{''}$ 的工序 A_i, 然后处理所有的 $J_i \in V$ 的工序 B_i (如图 4). 由式 (3) 可知, $\sum_{J_i \in V'} a_i \leq \sum_{J_i \in V''} a_i$, 因此在 M_2 上的处理过程中不会产生空闲时间, 由此可得

$$C_{\max}(S_3) = a_r + \sum_{J_i \in V''} a_i + \sum_{J_i \in V} b_i$$

不难看出 S_3 是 $S(PS_3)$ 的一个 ReSHFS 调度, 故由引理 3 可得

$$C_{\max}(S(PS_3)) \leq a_r + \sum_{J_i \in V''} a_i + \sum_{J_i \in V} b_i \tag{4}$$

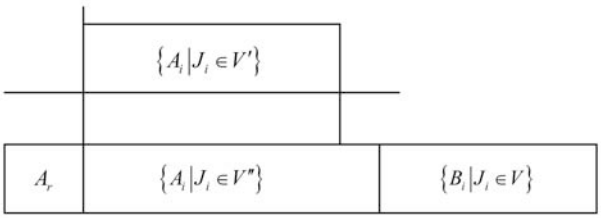

图 4 $S(PS_3)$ 的一个 ReSHFS 调度 S_3

Fig. 4 A ReSHFS schedule S_3 from $S(PS_3)$

接下来考虑 $S(PS_4)$. 同样先构造一个它的 ReSHFS 调度 S_4: 在 M_1 上, 先在 0 时刻开始处理所有 $J_i \in V^{'}$ 的工序 A_i, 接着处理 A_r; 在 M_2 上, 在 0 时刻开始处理所有 $J_i \in V^{''}$ 的工序 A_i, 接着处理所有 $J_i \in V\backslash\{J_r\}$ 的工序 B_i, 最后尽可能早地处理工序 B_r, 根据情况有两种可能 (如图 5). 与上面类似可知 S_4 是 $S(PS_4)$ 的 ReSHFS 调度, 且有 $C_{\max}(S(PS_4)) \leq C_{\max}(S_4)$.

由图 5 (a) 和 5 (b) 不难看出

$$C_{\max}(S_4) = \sum_{J_i \in V''} a_i + \sum_{J_i \in V} b_i$$

或

$$C_{\max}(S_4) = \sum_{J_i \in V'} a_i + a_r + b_r$$

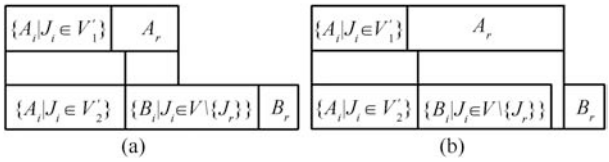

图 5 $S(PS_4)$ 的一个 ReSHFS 调度 S_4

Fig. 5 A ReSHFS schedule S_4 from $S(PS_4)$

1) 若 $C_{\max}(S_4) = \sum_{J_i \in V''} a_i + \sum_{J_i \in V} b_i$, 由式 (3)、引理 4 和情况 2 的条件 $\sum_{i=1}^{n} a_i > C_{\max}(S^*)$ 可知

$$\begin{aligned} C_{\max}(S(PS_4)) \leq C_{\max}(S_4) = & \\ \sum_{J_i \in V''} a_i + \sum_{J_i \in V} b_i \leq & \\ \frac{1}{2}\sum_{J_i \in V} a_i + \sum_{J_i \in V} b_i = & \\ \sum_{J_i \in V} (a_i + b_i) - \frac{1}{2}\sum_{J_i \in V} a_i \leq & \\ \left(2 - \frac{1}{2}\right) C_{\max}(S^*) = \frac{3}{2}C_{\max}(S^*) & \end{aligned}$$

2) 若 $C_{\max}(S_4) = \sum_{J_i \in V'} a_i + a_r + b_r$, 则

$$C_{\max}(S(PS_4)) \leq \sum_{J_i \in V'} a_i + a_r + b_r \tag{5}$$

由式 (4) 和式 (5), 可得

$$\begin{aligned} C_{\max}(S(PS_3)) + C_{\max}(S(PS_4)) \leq & \\ \left(a_r + \sum_{J_i \in V''} a_i + \sum_{J_i \in V} b_i\right) + & \\ \left(\sum_{J_i \in V'} a_i + a_r + b_r\right) = & \\ \sum_{J_i \in V} (a_i + b_i) + a_r + b_r & \end{aligned}$$

根据引理 4, 不难得出 $C_{\max}(S(PS_3)) + C_{\max}(S(PS_4)) \leq 3C_{\max}(S^*)$, 即 $C_{\max}(S(H)) \leq \min\{C_{\max}(S(PS_3)), C_{\max}(S(PS_4))\} \leq \frac{3}{2}C_{\max}(S^*)$.

下面的例子说明 3/2 的界是紧的. 考虑三个任务, 各工序的处理时间分别为: $a_1 = K$, $a_2 = K$, $a_3 = 2K$; $b_1 = 1$, $b_2 = 1$, $b_3 = 1$, 其中 $K \geq 1$.

最优调度 S^*: $V_1 = \{J_3\}, V_2 = \{J_1, J_2\}$, 即工序 (A_1, A_2) 依次在 M_1 上处理, 而 (A_3, B_3, B_1, B_2) 依次在 M_2 上处理. 此时最优目标值为 $C_{\max}(S^*) = 2K + 3$.

表 1　$C_{\max}(H)$ 与 $C_{\max}(GL)$ 的结果比较

Table 1　Comparison of $C_{\max}(H)$ and $C_{\max}(GL)$

Team	1 (15jobs)		2 (15jobs)		3 (15jobs)		4 (15jobs)		5 (15jobs)		6 (15jobs)		7 (15jobs)		8 (15jobs)		9 (15jobs)		10 (15jobs)	
	12 252	82 992	60 765	94 877	85 403	6 612	31 315	40 794	10 152	14 361	30 197	1 611	21 755	44 793	60 549	50 924	80 133	26 080	58 200	74 158
	90 532	12 784	78 292	41 074	90 326	98 878	58 922	93 048	10 000	8 009	82 714	87 019	1 871	14 624	42 378	21 824	74 605	20 300	20 341	57 159
	48 062	79 454	6 043	52 233	76 012	6 417	5 193	9 875	62 623	58 986	83 212	64 378	64 833	56 633	33 457	24 543	59 033	88 930	38 307	66 570
	7 846	65 781	42 280	22 311	16 127	62 556	71 082	5 438	10 390	55 831	87 128	37 143	43 484	4 985	72 978	62 645	79 009	63 747	77 885	61 165
	8 157	30 393	58 496	27 044	43 838	44 762	33 431	88 014	17 658	89 779	28 539	2 916	16 023	34 379	3 584	85 721	78 153	92 429	84 316	67 971
	38 792	48 951	60 273	33 795	31 429	32 471	94 356	54 058	9 836	58 270	57 406	19 492	22 616	86 296	32 806	21 739	4 717	16 730	3 444	73 693
	34 452	79 823	8 976	71 802	75 734	11 303	61 205	23 424	79 030	84 515	34 554	23 263	51 800	64 181	21 159	52 837	67 251	14 174	36 106	70 564
	25 935	75 905	86 773	6 460	35 543	58 451	35 398	38 608	74 235	5 683	88 107	15 280	4 158	53 611	38 541	65 639	77 020	24 046	97 439	93 022
	10 006	81 666	47 358	28 063	46 840	1 685	96 721	24 666	44 792	60 752	90 970	10 420	99 084	63 600	45 981	95 760	3 367	26 454	66 753	1 356
	30 750	74 815	85 052	87 446	22 707	71 507	49 438	2 424	67 198	1 688	17 652	32 655	68 738	93 540	84 325	94 658	51 519	83 872	94 025	37 363
	90 887	92 288	35 110	82 858	82 431	47 040	40 136	5 543	39 680	31 791	63 086	90 073	25 487	61 005	53 854	54 359	52 797	94 329	63 643	42 912
	79 553	66 805	74 054	7 804	68 560	42 172	8 357	26 101	47 152	67 961	40 635	96 612	15 616	71 736	94 641	49 875	51 447	26 977	68 813	81 867
	13 383	53 255	58 409	6 360	9 890	51 998	10 331	96 538	66 745	30 873	32 215	32 107	4 783	31 717	73 541	82 553	35 830	98 885	86 888	90 050
	17 202	42 896	37 302	87 285	95 543	19 825	85 287	65 279	25 254	91 911	2 972	71 072	28 256	38 578	23 688	1 236	14 147	17 355	58 905	36 687
	97 285	11 322	14 976	22 140	2 093	32 641	61 594	85 278	22 904	77 915	55 462	25 322	81 345	5 827	91 269	27 312	84 927	17 631	9 872	68 130
					88 115	75 311	26 381	69 357	80 395	35 712	24 420	62 106			23 868	38 745	22 953	10 591	20 045	13 635
					53 416	89 678	83 957	72 867	39 596	45 984	70 320	92 674			88 053	65 435	53 887	59 724	93 702	20 700
					82 604	89 642			83 159	9 651					60 813	91 155	51 547	98 543		
					22 641	74 109			29 359	95 753					81 447	37 387	75 448	97 558		
															35 283	52 157				
$C_{\max}(H)$	906 976		760 519		1 030 937		855 528		935 261		891 200		727 376		1 080 088		1 028 381		980 040	
$C_{\max}(GL)$	1 095 505		933 034		1 266 694		1 114 397		1 220 951		1 070 837		919 867		1 441 339		1 326 515		1 301 739	

不难计算, 算法 H 的四个调度的目标值分别为 $4K+3$, $4K+1$, $3K+3$ 和 $3K+1$, 由此可得 $C_{\max}(S(H))=3K+1$. 因此, 当 $K\to\infty$ 时,

$$\frac{C_{\max}(H)}{C_{\max}(S^*)}=\frac{3K+1}{2K+3}\to\frac{3}{2}$$

□

2.2　数值结果分析

我们随机挑选了 10 个任务集, 任务数在 15～20 之间, 工序的处理时间随机的从 1 到 100 000 进行选取. 记原有贪婪算法[12] 的目标值为 $C_{\max}(GL)$, 算法 H 的目标值为 $C_{\max}(H)$. 从表 1 的结果不难得出, 平均情况看, $C_{\max}(H)$ 值比 $C_{\max}(GL)$ 的值要小 20 % 左右. 由此可见我们的算法无论从绝对性能方面还是实际计算情况来看都大大地改进了原来的算法.

3　结论

本文主要研究了一类图形处理中的混合 Flow-shop 调度问题, 设计了一个多项式时间的改进算法, 主要思想是结合平行机调度问题的算法, 在任务处理模式方面引入了四种划分方式并从中选取最好的一种作为结果输出. 主要结论有: 该算法的最坏情况界为 3/2, 改进了原有的 8/5 的界; 在实际例子的效果方面, 本算法的目标值比原有的贪婪算法的结果要小 20 % 左右, 因此, 从绝对性能和实际效果都大大改进了已有的结果.

References

1 Wei Q, He Y. A two-stage semi-hybrid flowshop problem in graphics processing. *Applied Mathematics — A Journal of Chinese Universities*, 2005, **20**(4): 393−400

2 Kouvelis P, Vairaktarakis G. Flowshops with processing flexibility across production stages. *IIE Transactions*, 1998, **30**(8): 735−746

3 Vairaktarakis G, Lee C Y. Analysis of algorithms for two-stage flowshops with multi-processor task flexibility. *Naval Research Logistics*, 2004, **51**(1): 44−59

4 Boudhar M, Meziani N. Two-stage hybrid flow shop with recirculation. *International Transactions in Operational Research*, 2010, **17**(2): 239−255

5 Besbes W, Teghem J, Loukil T. Scheduling hybrid flow shop problem with non-fixed availability constraints. *European Journal of Industrial Engineering*, 2010, **4**(4): 413−433

6 Wang Wan-Liang, Yao Ming-Hai, Wu Yun-Gao, Wu Qi-Di. Hybrid flow-shop scheduling approach based on genetic algorithm. *Journal of System Simulation*, 2002, **14**(7): 863−869 (王万良, 姚明海, 吴云高, 吴启迪. 基于遗传算法的混合 Flow-shop 调度方法. 系统仿真学报, 2002, **14**(7): 863−869)

7 Oguz C, Ercan M F. A genetic algorithm hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling*, 2005, **8**(4): 323−351

8 Zhang Y, Li X P, Wang Q. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. *European Journal of Operational Research*, 2009, **196**(3): 869−876

9 Zhan Y, Qiu C H, Xue K. A hybrid genetic algorithm for hybrid flow shop scheduling with load balancing. *Key Engineering Materials*, 2009, **392−394**: 250−255

10 Hu Rong, Qian Bin. A hybrid differential evolution algorithm for stochastic flow shop scheduling with limited buffers. *Acta Automatica Sinica*, 2009, **35**(12): 1580−1586 (胡蓉, 钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法. 自动化学报, 2009, **35**(12): 1580−1586)

11 Gupta J N D, Koulamas C P, Kyparisis G J, Potts C N, Strusevich V A. Scheduling three-operation jobs in a two-machine flow shop to minimize makespan. *Annals of Operations Research*, 2004, **129**(1−4): 171−185

12 Wei Qi, Jiang Yi-Wei. Approximation algorithms for a two-stage hybrid flow shop. *Journal of Software*, to be published

13 Johnson S M. . *Naval Research Logistics Quarterly*, 1954, **1**(1): 61−68

14 Graham R L. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 1966, **45**(9): 1563−1581

蒋义伟 浙江理工大学理学院副教授. 2007 年获浙江大学数学系博士学位. 主要研究方向为组合优化, 调度理论, 算法设计与分析. 本文通信作者.
E-mail: mathjyw@yahoo.com.cn
(**JIANG Yi-Wei** Associate professor at the Faculty of Science, Zhejiang Sci-Tech University. He received his Ph. D. degree from the Department of Mathematics, Zhejiang University in 2007. His research interest covers combinatorial optimization, scheduling, design and analysis of algorithm. Corresponding anthor of this paper.)

魏 麒 浙江大学宁波理工学院讲师. 2006 年获浙江大学数学系硕士学位. 主要研究方向为组合优化和调度理论.
E-mail: weiqi@nit.zju.edu.cn
(**WEI Qi** Lecturer at Ningbo Institute of Technology, Zhejiang University. He received his master degree from the Department of Mathematics, Zhejiang University in 2006. His research interest covers combinatorial optimization and scheduling.)