第 41 卷 第 3 期 应 用 数 学 学 报 Vol. 41 No. 3
2018 年 5 月 ACTA MATHEMATICAE APPLICATAE SINICA May, 2018

一类带平行机的两阶段柔性流水调度近似算法*

张明会

(大连理工大学软件学院软件工程系，大连 116620)

(大连东软信息学院软件工程系，大连 116023)

(E-mail: neusoftzhang@163.com)

韩 鑫

(大连理工大学软件学院软件工程系，大连 116620)

摘 要 本文研究一类柔性流水调度与平行机调度相结合的两阶段流水调度模型，模型中第 1 阶段有 1 台机器，第 2 阶段有 m 台同构并行机，每个任务在第 2 阶段需要 size_i 台机器同时并行执行. 目标是所有任务都完成的完工时间最小化. 该模型已被证明出是强 NP 难的，并给出了在某种特定情况下近似比为 3 的近似算法. 本文首先详细分析了前人近似算法基本过程，给出该算法近似比分析的局限性；接着给出了一个近似比为 3 的算法，摒弃了前人给出的近似比为 3 时的约束条件；最后研究了当第 2 阶段机器数为 2 和 3 时的两种特定情况，采用列表调度思想，给出了近似比为 2.5 和 2.67 的近似算法.

关键词 柔性流水调度；平行机调度；近似算法；近似比

MR(2000) 主题分类 03f99

中图分类 O224.10

1 引言

调度问题通常是给定一组具有一定特征的任务，在时间维度上为各个任务分配有限资源 (如处理机), 以期满足一个或多个目标 (如最小化最大完工时间). 调度一般情况下是一个优化问题，时间和资源是问题的核心，给出一个调度时间表可能比较容易，但

本文 2017 年 12 月 31 日收到. 2018 年 4 月 5 日收到修改稿.

* 国家自然科学基金 (11570160,11701062), 辽宁省自然科学基金 (20170540050) 和大连市高层次人才创新支持计划 (2015R095) 资助项目.

如何获得一个最优时间表就比较困难.

许多工业生产系统都是一个流水作业环境，包括多个阶段的生产流程，所有的作业必须以同样的顺序经过这些阶段. 经典的流水调度中每个阶段仅包括一台作业机器，在同一时刻只能执行一个操作. 本文研究一类特殊的两阶段流水调度模型： 每一任务必须在第 **1** 阶段完成后才能开始第 **2** 阶段工作，第 **1** 阶段只有 **1** 台机器，第 **2** 阶段有 m 台同构并行机，每个任务在第 **2** 阶段需要多台机器同时开始执行，且在这多台机器上执行时间相同，所有操作在执行过程中不允许中断. 从各阶段机器数特征看，该模型属于 **柔性流水调度 (flexible flow shop, FFS)** 范畴，特殊性在于通用 FFS 模型中每个任务在每个阶段都只需要一台机器，而本文研究模型任务在第 2 阶段需要多台机器；从第 2 阶段的机器特征及任务调度特征看，又与 **平行机调度 (parallel scheduling)** 相关. 关于这两种调度模型独立的研究成果已有许多，但将这两种模型相结合的研究较少.

柔性流水调度：Arthanari, Ramamurthy 和 Salvador 作为研究 FFS 问题的先驱在 [1] 和 [2] 中首先定义了 FFS 问题，他们将通用的柔性流水调度模型定义为：有 k 个阶段，至少有一个阶段有多台机器，其它阶段允许只有一台机器，问题是如何在每个阶段将多个任务分配到一台或多台机器上，并且为分配到同一台机器上的任务作好时间排序，目的是使某一衡量指标最小化. 自 FFS 调度问题被正式定义以来，众多学者已经展开了系列的研究 [3−7]. 然而，由于经典 FFS 模型中的若干假设条件在实际应用中很难实现，使得对经典的 FFS 问题研究饱受争议. 结果众多学者开始研究基于经典 FFS 模型而衍生出来的若干 FFS 变型问题，如 Chun-Lung 和 Chuen-Lung 等人提出的各阶段非相关并行机的多阶段 FFS 调度问题 [8], Yazid 等人考虑生产实际中带块约束的 FFS 调度问题等 [9]. 关于 **两阶段 FFS 调度模型**, 根据约束条件与调度环境不同，涌现出众多研究模型及研究结果. 如 Wang Hui[10] 和 Tian[11] 提出以释放时间为研究目标的两阶段 FFS 问题， Li Zhan-tao 等人研究了考虑任务结束时间限制以最小化任务延迟时间为研究目标的两阶段 FFS 调度问题 [12] 等. 2011 年，作为大数据环境下流行的批数据处理框架 Map-Reduce, 被 UIUC 大学的 Moseley 和 Yahoo 研究院的 Dasgupta 等人归结为一类两阶段并行任务调度问题，并给出近似度为 12 的近似算法 [13]. Byung-Cheon Choi 和 Kangbok Lee 等人研究了第 1 阶段单机第 2 阶段 m 台同型机，每个任务在两个阶段的处理时间相同的两阶段柔性流水调度情况，以最小化完工时间为研究目标，给出了一个近似算法，当机器数 $m = 2$ 时，算法能获得 $\frac{5}{4}$ 的近似比，当 $m \geq 3$ 时，算法能获得 $\frac{\sqrt{1+m^2}+1+m}{2m}$ 的近似比 [14]. 2014 年， [15] 综合考虑 GPU 系统中大规模并行负载同时进行数据传输和加载情况，以总负载 makespan 最小化作为系统性能的全局优化目标，将负载 “传输 - 执行” 联合调度问题建模成两阶段流水调度问题，基于贪婪策略给出了在某种特定条件下近似度为 3 的多项式时间近似算法，从理论层面上证明了 GPU 系统两阶段调度的有效性.

平行机调度：平行机调度亦是调度领域基本模型之一，其数学模型可表述为：有一包含 n 个任务的任务列表 $L = (J_1, J_2, \cdots, J_n)$, 每个任务 J_i 有一个执行 / 处理时间 p_i 及事先给定的所需机器数 size_i, 可供使用的机器总数为 m. 调度序列形如 $S = ((s_1, r_1), (s_2, r_2), \cdots, (s_n, r_n))$, 其中 s_i 表示任务 J_i 的开始加工时间， $r_i \subseteq \{1, 2, \cdots, m\}$

表示任务 J_i 所使用的机器编号集合，即 $|r_i| = \text{size}_i$. 若某调度序列 S 能够保证同一时刻任何一台机器至多处理一个任务 (工件), 则称该调度序列 S 是可行的，该调度的长度定义为最后被完成的任务的完工时间，即 $Cmax = \max\{s_i + p_i | i \in \{1, 2, 3, \cdots, n\}\}$. 平行机调度一般环境下，任务 J_i 分配的机器编号没有要求，然而在一些特定情况下，要求所分配的机器编号必须连续. 利用 [16] 中关于平行机调度问题的三段式表示法，本文中将 **无机器编号连续** 需求的平行机调度问题表述为 $Pm|\text{size}_i|C\max$. Du 等人在 [17] 中已经证明当可用机器总数为 5 时模型 $P5|\text{size}_i|C\max$ 是强 NP 难的，因此 $Pm|\text{size}_i|C\max$ 亦是强 NP 难的， [18] 中证明出没有任何近似算法能得到 1.5 倍近似比，除非 P=NP. [19] 和 [20] 指出当可用机器数是一个常数时， $Pm|\text{size}_i|C\max$ 存在多项式时间近似方案 (PTAS). 关于一般情况下的平行机调度问题近似算法研究，最初是受到资源约束调度问题结果的启发， 1975 年 Garey 和 Graham 等人提出了一个关于资源约束调度问题近似比为 2 的近似算法 [21], 在该调度环境下，给定一个或多个资源，每个任务需要一定数量的每个资源一个或多个. Ludwig 和 Tiwari 在 [22] 中指出，若将 $Pm|\text{size}_i|C\max$ 环境中的可用处理器看作唯一的一类资源，进而得出该算法解决 $Pm|\text{size}_i|C\max$ 调度问题，能得到近似比为 2, 本文将其称为引理 1.

引理 1 用 Garey, Graham 算法解决 $Pm|\text{size}_i|C\max$ 调度问题，得到近似比为 2.

列表调度 (list scheduling) 算法是平行机调度中任务仅需要一台机器时的经典算法. 其基本思想是按照优先级依次安排至可用空闲机器上加工，该算法执行时最坏情况即为加工时长最大的那个任务在最后加工，且其开始加工时刻其它机器均无加工任务，此时不难得出如下引理 2.

引理 2 采用列表调度思想对 n 个任务在 m 台机器上进行平行机调度，若任务 τ_i 的执行时间为 p_i, 则调度的最大完工时间一定满足 $C\max \le \frac{1}{m}\sum\limits_i p_i + \left(1 - \frac{1}{m}\right)\max\limits_i\{p_i\}$.

本文贡献：本文将流水调度与平行机调度相结合，研究一类第 1 阶段单机第 2 阶段 m 台并行机的两阶段流水调度问题，文中首先分析了孙等人在 [15] 中给出的近似算法的局限性；接着用 Garey, Graham 算法给出一个近似比为 3 的近似算法，该算法分析时摒弃了孙等人的约束条件；最后给出当第 2 阶段并行机器数分别是 2 和 3 特定情况下的两个近似算法，并证明其近似比分别为 2.5 和 2.67.

本文余下章节如下：第 2 节对欲研究的问题给出具体描述及前人结果. 第 3 节给出了一个近似比为 3 的近似算法. 第 4 节给出该模型两种特殊情况下的近似算法. 最后总结全文，并对未来值得关注的研究方向进行初步探讨.

2 问题描述与应用

2.1 问题描述与建模

本文所研究的这类两阶段流水调度问题，可详细描述为：给定一包含 n 个任务的任务集 $\Gamma = (\tau_1, \tau_2, \cdots, \tau_n)$ 及一个两阶段 flow-shop 流水调度环境，第 1 阶段中只有 1 台机器，第 2 阶段有 m 台同构并行机，任务 τ_i 可用三元组 $\{p_{1i}, \text{size}_i, p_{2i}\}$ 来表示，其中各参数含义如下：

- p_{1i} 为任务 τ_i 在第 1 阶段执行时间；
- size_i 为任务 τ_i 在第 2 阶段需要的并行机数量；
- p_{2i} 为任务 τ_i 在第 2 阶段的执行时间；

该模型要求任务 τ_i 的第 2 阶段操作必须在第 1 阶段完成后，由第 2 阶段 m 台机器中的 size_i 台空闲机器同时并行开始，每台机器上的执行时间均为 p_{2i}, 这 size_i 台机器 **不必连续**, 所有操作在执行过程中不允许抢占，即操作一旦开始，在它完成之前不允许中断. 问题的优化目标是所有负载任务完成的时间最短，完工时间定义为 $C\max$, 该问题在 [15] 中已被证明是 NP-hard 问题. 根据调度问题的三参数表示法 [23], 可记为 $F2(1,P_m)|\text{size}_i|C\max$, 其中：

- $F2(1,P_m)$ 表示两阶段 flow-shop 调度环境，阶段 1 只有 1 台机器，阶段 2 的有 m 台同构并行机；
- size_i 代表并行任务集合，即任意任务 τ_i 包含 size_i 个操作；
- $C\max$ 代表问题的研究目标，即最小化最大完工时间.

根据上述定义知, 若所有任务第 2 阶段只需要 1 台机器，则该模型即为通用的两阶段 FFS 调度模型，即表明两阶段 FFS 调度属于 $F2(1,P_m)|\text{size}_i|C\max$ 的特例情况.

2.2 模型应用与前人结果

在 GPU 系统中，并行计算平台中负载调度策略的好坏是衡量其计算性能的主要因素，但考虑数据传输时长的 GPU 负载调度还处于较低的自动化水平， [15] 综合考虑 GPU 独享内存与主存之间的数据传输延时问题，将 $F2(1,P_m)|\text{size}_i|C\max$ 模型应用至 GPU 负载调度，以此优化 GPU 中的通信和计算资源的统筹分配.

[15] 中将负载的时间信息和并行任务数与矩形域的二维空间联系起来，建立负载的 2D 双层矩形域模型，由此将 GPU 上负载调度问题归结为一类新的双层 Strip-Packing 问题，用负载的运输时间和执行时间刻画矩形的高度，矩形的空白域表示运输，网格域表示执行，矩形的宽度表示负载执行阶段所用的机器数，负载与矩形的转换如下图 1 所示.

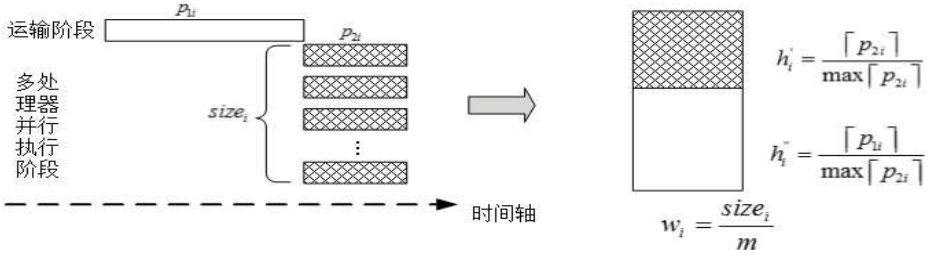

图 1　负载特征与矩形转换过程

按上述转换过程，任务集 Γ 中的 n 个任务转换为矩形列表 L 中的 n 个矩形，该 2D 双层 Strip-packing 问题的研究目标是如何将 L 中的 n 个矩形放入宽度为 1 高度不限的带状区域中，要求每个矩形的网格域均在空白域之上，所有矩形网格域不能相交重叠，任一水平线最多与 1 个空白域相交，目的是如何放置使得所用到的带状区域高度最小.

根据模型的转换过程知，该 2D 双层 Strip-packing 的可行解一定能得到两阶段流水调度问题 $F2(1,P_m)|\text{size}_i|C\max$ 的一个可行调度序列，[15] 中给出一个求 2D 双层 Strip-packing 问题可行解的算法 A, 其基本过程是先按照矩形 $w_i\times h_i'/h_i''$ 非递增的顺序依次放置空白域，再根据不同情况，将条纹域回填至空白域，由此得到 Strip-packing 问题的最优松弛解，最后通过一定机制将该松弛解扩张得到可行解. 同时证明得出该最优松弛解即为 2D 双层 Strip-packing 问题的下界 (记为 $\text{OPT}_{\text{low}}(L)$), 该文献中已求得下界值

$$\text{OPT}_{\text{low}}(L)=\max\Big\{\sum_{i=1}^{n}h_i',\ \sum_{i=1}^{n}w_i\times h_i''\Big\}.$$

关于算法 A 的近似比分析，[15] 给出了如下结论：

1) 若 $\text{OPT}_{\text{low}}(L)=\sum\limits_{i=1}^{n}\{w_i\times h_i''\}$, 则算法 A 存在伪多项式 $3\sum\limits_{i=1}^{n}w_i$ 的上界；

2) 若 $\text{OPT}_{\text{low}}(L)=\sum\limits_{i=1}^{n}h_i'$, 且任务同时满足条件 $\min\limits_{1\le i\le n}\lceil p_{1i}\rceil\ge\max\limits_{1\le i\le n}\lceil p_{2i}\rceil$ 时，算法 A 具有常近似度 3.

对于 $\text{OPT}_{\text{low}}(L)=\sum\limits_{i=1}^{n}h_i'$ 的情况，若存在某一任务，其第 1 阶段的执行时长小于第 2 阶段的执行时长，则算法 A 能否得到 3 的近似度文中并未做详细解释或说明，因此文中给出近似度为 3 的多项式时间近似算法结论是不严密的. 同时，由于问题描述中未指明任务所需的 size_i 台机器是否必须连续，因此 2D 双层 Strip-packing 的最优解并不等同于 $F2(1,P_m)|\text{size}_i|C\max$ 的最优解，在算法分析过程中不能将其完全等价.

[15] 所提出解决 $F2(1,P_m)|\text{size}_i|C\max$ 模型的近似算法并不能保证任何时候都能达到 3 的近似度，鉴于此，本文将调度的两个阶段分开考虑，借鉴已有的平行机调度算法，从理论上提出一个不需要任何约束条件的近似度为 3 的近似算法.

3 带平行机的两阶段柔性流水调度近似算法

3.1 近似算法

本文所研究的 $F2(1,P_m)|\text{size}_i|C\max$ 两阶段流水调度模型的第 2 阶段有明显的机器并行特征，鉴于目前平行机调度问题研究成果较多，故本文给出一个先将所有任务在第 1 阶段执行完毕，再进行平行机调度的近似算法 A1. 设 OPT (Γ) 表示最优算法求得的问题 $F2(1,P_m)|\text{size}_i|C\max$ 的最小完工时间. 算法具体过程如下：

算法 **A1:** $F2(1,P_m)|\text{size}_i|C\max$ 近似算法

输入: 负载任务集合 Γ 及 m 台并行处理器;

输出: 一个满足最大完工时间 $C\max \le 3\text{OPT}(\Gamma)$ 的调度序列;

过程:

BEGIN

1. 将任务集 Γ 中所有任务按随机顺序安排在第 1 阶段的这一台机器上，期间不间断，一个任务执行完毕后立即开始下一任务;
2. 将所有任务作为平行机调度 $Pm|\text{size}_i|C\max$ 的输入列表，按照 Garey&Graham 算法完成调度.

END

3.2 算法分析

定理 1 算法 A1 的近似比为 3.

证 为算法分析方便，现定义如下几个符号:

- S: 对输入序列 Γ, 算法 A1 求得问题 $F2(1,P_m)|\text{size}_i|C\max$ 的可行调度序列;
- S^p: 对输入序列 Γ, Garey, Graham 算法求得平行机调度 $Pm|\text{size}_i|C\max$ 的可行调度序列;
- $C\max(S)$: 算法 A1 求得问题 $F2(1,P_m)|\text{size}_i|C\max$ 的可行调度序列 S 的最小完工时间;
- $C\max(S^p)$: Garey, Graham 算法求得平行机调度 $Pm|\text{size}_i|C\max$ 的可行调度序列 S^p 的最小完工时间;
- $\text{OPT}(\Gamma)^p$: 最优算法求得的问题 $Pm|\text{size}_i|C\max$ 的最小完工时间;

对同一输入任务列表 Γ 而言，问题 $F2(1,P_m)|\text{size}_i|C\max$ 的最优解最小完工时间一定大于问题 $Pm|\text{size}_i|C\max$ 的最优解最小完工时间 $\text{OPT}(\Gamma)^p$, 如若不然，由于每一任务第 1 阶段的执行时间 p_{1i} 都是个非负数，则按问题 $F2(1,P_m)|\text{size}_i|C\max$ 的最优解中 m 台并行机上序列完成 $Pm|\text{size}_i|C\max$ 调度所用到的完工时间一定比 $\text{OPT}(\Gamma)^p$ 还要少，这与 $\text{OPT}(\Gamma)^p$ 是最优完工时间相矛盾，因此一定有下式成立:

$$\text{OPT}(\Gamma)^p \le \text{OPT}(\Gamma). \tag{1}$$

根据算法 A1 的执行过程知， A1 的最小完工时间 $C\max(S)$ 一定满足下式:

$$C\max(S) = \sum_{i=1}^{n} p_{1i} + C\max(S^p). \tag{2}$$

由于第 1 阶段中的这台机器每次只能执行一个任务，且流水环境要求所有任务的第 2 阶段操作一定在第 1 阶段完成后开始，故有：

$$\sum_{i=1}^{n} p_{1i} \leq \mathrm{OPT}\,(\Gamma). \tag{3}$$

根据引理 1 的结论知：

$$C\max(S^p) \leq 2\mathrm{OPT}\,(\Gamma)^p. \tag{4}$$

结合式子 (1)–(4) 有：

$$C\max \leq \sum_{i=1}^{n} p_{1i} + 2\mathrm{OPT}\,(\Gamma)^p \leq \mathrm{OPT}\,(\Gamma) + 2\mathrm{OPT}\,(\Gamma) = 3\mathrm{OPT}\,(\Gamma). \tag{5}$$

由此得出算法 A1 的近似比为 3, 定理 1 得证. 证毕.

4 两种特殊情况下的近似算法

算法 A1 虽能得到近似比为 3 的结果, 但 Garey, Graham 算法原本是用来解决资源约束调度问题, 应用至平行机调度上时实现过程有些繁杂, 考虑到实际生产生活中第 2 阶段并行机数量存在常数情况, 本节我们考察并行机数为 2 和为 3 时的两种特定情况, 这里分别表示为 $F2(1,P_2)|\mathrm{size}_i|C\max$ 和 $F2(1,P_3)|\mathrm{size}_i|C\max$, 用 [24] 中的方法能够证明当所有任务的 $\mathrm{size}_i = 1$ 时即模型 $F2(1,P_2)||C\max$ 是强 NP 难的, 因此 $F2(1,P_2)|\mathrm{size}_i|C\max$ 亦是强 NP 难的, 同理, 模型 $F2(1,P_3)|\mathrm{size}_i|C\max$ 亦是强 NP 难的. 下面本文将给出两个实现过程比 Garey, Graham 更简单且近似比小于 3 的近似算法.

4.1 第 2 阶段并行机数量为 2 时的近似算法 A2

对于本文所研究的两阶段负载调度模型, 若第 2 阶段的并行机器数量为 2, 按调度问题三段式表示法可表示为 $F2(1,P_2)|\mathrm{size}_i|C\max$. 本文根据任务在第 2 阶段需要 1 台或 2 台平行机的不同情况做不同机器分配, 按照 list-scheduling 思想给出 $F2(1,P_2)|\mathrm{size}_i|C\max$ 问题的近似算法 A2, 算法中用 Γ_1 表示第 2 阶段需要 1 台并行机的所有任务集， n_1 表示 Γ_1 中任务数； Γ_2 表示第 2 阶段需要 2 台并行机的所有任务集， n_2 表示 Γ_2 中任务数，算法 A2 的具体执行过程如下.

算法过程

算法 **A2:** $F2(1,P_2)|\text{size}_i|C\max$ 近似算法

输入: 负载任务集合 Γ 及 2 台并行处理器;

输出: 任务集 Γ 中任务的执行次序;

过程:

BEGIN

1. 令 Γ_1 是第 2 阶段仅需要一台机器的任务的集合，令 Γ_2 是第 2 阶段仅需要两台机器的任务的集合;
2. 将任务集 Γ 中所有负载任务按随机顺序依次放在第 1 阶段的一台机器上执行，期间不间断，即完成一个负载后立即开始执行下一负载;
3. 将集合 Γ_2 中所有负载任务随机依次放在这两台并行机上执行;
4. 将集合 Γ_1 中所有负载任务按 list scheduling 原则放在这两台并行机上执行.

END

算法分析

定理 2 算法 A2 的近似比为 2.5.

证 设问题 $F2(1,P_2)|\text{size}_i|C\max$ 的最优解的最小完工时间为 $\text{OPT}(\Gamma)$, 则有下式 (6)–(8) 成立:

$$\text{OPT}(\Gamma) \geq \sum_{i=1}^{n} p_{1i}, \tag{6}$$

$$\text{OPT}(\Gamma) \geq \sum_{\tau_i \in \Gamma_2} p_{2i} + \frac{1}{2} \sum_{\tau_i \in \Gamma_1} p_{2i}, \tag{7}$$

$$\text{OPT}(\Gamma) \geq \max_{i=1}^{n} p_{2i}. \tag{8}$$

设算法 A2 的完工时间为 $C\max$, 则根据算法 A2 的执行过程和引理 2, 能够得出式 (9):

$$C_{\max} \leq \sum_{i=1}^{n} p_{1i} + \sum_{\tau \in \Gamma_2} p_{2i} + \frac{1}{2} \sum_{\tau \in \Gamma_1} p_{2i} + \frac{1}{2} \max_{1 \leq i \leq n} \{p_{2i}\}. \tag{9}$$

结合式 (6)–(9) 有:

$$C_{\max} \leq \text{OPT}(\Gamma) + \text{OPT}(\Gamma) + \frac{1}{2}\text{OPT}(\Gamma) = 2.5\text{OPT}(\Gamma), \tag{10}$$

即得出算法 A2 的近似比为 2.5, 定理 2 得证. 证毕.

4.2 第 2 阶段并行机数量为 3 时的近似算法 A2

与算法 A2 的思想类似，对第 2 阶段的并行机器数为 3 的特定情况，根据任务在第 2 阶段需要 1 台， 2 台和 3 台的不同情况，进行不同处理，按照 list-scheduling 思想给出 $F2(1,P_3)|\text{size}_i|C\max$ 问题的近似算法 A3, 类似地，用 Γ_1 表示第 2 阶段需要 1 台并行机的所有任务集， n_1 表示 Γ_1 中任务数； Γ_2 表示第 2 阶段需要 2 台并行机的所有任务

集， n_2 表示 Γ_2 中任务数； Γ_3 表示第 2 阶段需要 3 台并行机的所有任务集， n_3 表示 Γ_3 中任务数. 三台机器编号分别为 m_1, m_2 和 m_3. 算法 A3 的具体过程如下.

算法过程

算法 **A3:** $F2(1,P_3)|\text{size}_i|C\max$ 近似算法

输入: 负载任务集合 Γ 及 3 台并行处理器;

输出: 任务集 Γ 中任务的执行次序;

过程:

BEGIN

1. 令 $\Gamma_i(i=1,2,3)$ 是第 2 阶段仅需要 i 台机器的任务集合;
2. 将任务集 Γ 中所有负载任务按随机顺序依次放在第 1 阶段的第一台机器上执行，期间不间断，即完成一个负载任务后立即执行下一个;
3. 将集合 Γ_3 中所有负载任务随机依次放在这 3 台并行机上执行;
4. 将集合 Γ_2 中所有负载任务依次放在 m_1 和 m_2 两台机器上运行，将集合 Γ_1 中所有负载任务放在 m_3 机器上运行，此时分为两种情况:

   (1) 若 $\sum\limits_{\tau_i\in\Gamma_2} p_{2i} \le \sum\limits_{\tau_i\in\Gamma_1} p_{2i}$, 则 Γ_2 中所有任务均执行完成后， m_1 和 m_2 机器空闲，此时将 Γ_1 中剩余任务按 list scheduling 算法继续在这 m_1, m_2 和 m_3 上并行执行;

   (2) 若 $\sum\limits_{\tau_i\in\Gamma_2} p_{2i} > \sum\limits_{\tau_i\in\Gamma_1} p_{2i}$, 则 Γ_1 中所有任务均执行完成后， m_3 机器空闲，此时将 Γ_2 中剩余任务继续在这 m_1 和 m_2 上并行执行.

END

算法分析

定理 3 算法 A3 的近似比为 2.67.

证 设问题 $F2(1,P_2)|\text{size}_i|C\max$ 的最优解的最小完工时间为 $\text{OPT}(\Gamma)$, 算法 A3 的完工时间为 $C\max$, 由于执行阶段有两种情况，下面分别讨论:

情况 1 若 $\sum\limits_{\tau_i\in\Gamma_2} p_{2i} \le \sum\limits_{\tau_i\in\Gamma_1} p_{2i}$, 此时有:

$$\text{OPT}(\Gamma) \ge \sum_{i=1}^{n} p_{1i}, \tag{11}$$

$$\text{OPT}(\Gamma) \ge \frac{1}{3}\sum_{\tau_i\in\Gamma_1} p_{2i} + \frac{2}{3}\sum_{\tau_i\in\Gamma_2} p_{2i} + \sum_{\tau_i\in\Gamma_3} p_{2i}, \tag{12}$$

$$\text{OPT}(\Gamma) \ge \max_{i=1}^{n} p_{2i}. \tag{13}$$

为表述方便, 将 Γ_1 中任务标识为 τ_i^1 $(1\le i\le n_1)$, Γ_2 中任务标识为 τ_i^2 $(1\le i\le n_2)$, Γ_3 中任务标识为 τ_i^3 $(1\le i\le n_3)$. τ_k^1 为 Γ_2 中任务全部执行完毕的 $t3$ 时刻正在处理中的 Γ_1 中任务，图 2 描述了此种情况下算法执行过程.

此时根据算法 A3 执行过程知

$$t1 = \sum_{i=1}^{n} p_{1i},$$
$$t2 = t1 + \sum_{\tau_i \in \Gamma_3} p_{2i},$$
$$t3 = t2 + \sum_{\tau_i \in \Gamma_2} p_{2i}.$$

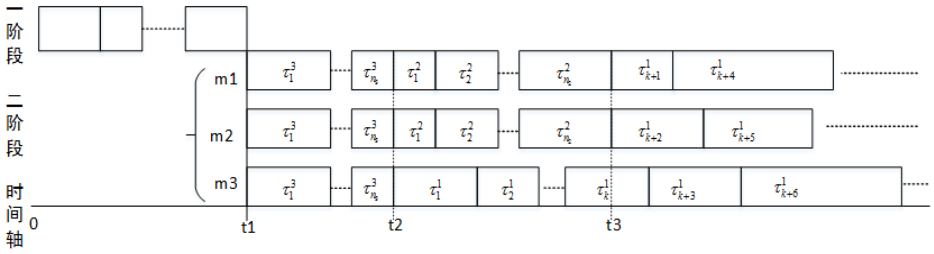

图 2　算法 A3 执行过程示意图

在 $t3$ 时刻之后任务的总剩余执行时间为 $\sum\limits_{\tau_i \in \Gamma_1} p_{2i} - \sum\limits_{\tau_i \in \Gamma_2} p_{2i}$. 此时将 Γ_3 中剩余任务按列表调度算法过程安排到 $m1$, $m2$ 和 $m3$ 上，根据引理 2, 不难得出 τ_k^1 在 $t3$ 时刻残余部分与 Γ_3 中剩余任务总的完工时间不会超过 $\frac{1}{3}\Big(\sum\limits_{\tau_i \in \Gamma_1} p_{2i} - \sum\limits_{\tau_i \in \Gamma_2} p_{2i}\Big) + \frac{2}{3} \max\limits_{1 \le i \le n} \{p_{2i}\}$.

因此算法 A3 的总完工时间一定满足

$$\begin{aligned} C_{\max} &\le \sum_{i=1}^{n} p_{1i} + \sum_{\tau_i \in \Gamma_3} p_{2i} + \sum_{\tau_i \in \Gamma_2} p_{2i} + \frac{1}{3}\Big(\sum_{\tau_i \in \Gamma_1} p_{2i} - \sum_{\tau_i \in \Gamma_2} p_{2i}\Big) + \frac{2}{3} \max_{1 \le i \le n} \{p_{2i}\} \\ &= \sum_{i=1}^{n} p_{1i} + \sum_{\tau \in \Gamma_3} p_{2i} + \frac{2}{3} \sum_{\tau \in \Gamma_2} p_{2i} + \frac{1}{3} \sum_{\tau \in \Gamma_1} p_{2i} + \frac{2}{3} \max_{1 \le i \le n} \{p_{2i}\}. \end{aligned} \tag{14}$$

结合式 (11)–(14) 有

$$C_{\max} \le \mathrm{OPT}(\Gamma) + \mathrm{OPT}(\Gamma) + \frac{2}{3}\mathrm{OPT}(\Gamma) = 2.67\mathrm{OPT}(\Gamma). \tag{15}$$

情况 2　若 $\sum\limits_{\tau_i \in \Gamma_2} p_{2i} < \sum\limits_{\tau_i \in \Gamma_1} p_{2i}$, 此时有

$$\mathrm{OPT}(\Gamma) \ge \sum_{i=1}^{n} p_{1i}, \tag{16}$$
$$\mathrm{OPT}(\Gamma) \ge \sum_{\tau_i \in \Gamma_3} p_{2i} + \sum_{\tau_i \in \Gamma_2} p_{2i}. \tag{17}$$

此时算法 A3 求得的最小完工时间满足

$$C_{\max} \leq \sum_{i=1}^{n} p_{1i} + \sum_{\tau_i \in \Gamma_3} p_{2i} + \sum_{\tau_i \in \Gamma_2} p_{2i}. \tag{4.18}$$

结合式 (16)–(18) 有

$$C_{\max} \leq \mathrm{OPT}(\Gamma) + \mathrm{OPT}(\Gamma) = 2\mathrm{OPT}(\Gamma). \tag{4.19}$$

综合式 (15) 和 (19) 知算法 A3 的近似比为 2.67, 定理 3 得证. 证毕.

5 总结

本文研究两阶段流水调度问题, 将经典的多阶段流水调度和并行调度问题相结合, 给出了一个近似比为 3 的近似算法, 该算法在分析时去掉了前人在近似比 3 分析时的约束条件. 最后本文又研究了当执行阶段机器数为 2 和 3 时的两种特定情况, 按照 list scheduling 基本思想, 给出两个近似算法, 分析其近似比分别为 2.5 和 2.67. 这两个算法执行过程比较简单, 在生产实际中实现起来比较容易.

本文虽然在近似比分析时去掉了约束条件, 但近似比并未有所改进, 对该问题仍有进一步的研究空间:

1) 提出更好的算法, 改进问题的近似比;

2) 改进孙等人提出的问题的下界;

3) 对执行阶段连续机器需求的模型进行独立研究, 获得更好的结果.

参 考 文 献

[1] Arthanari T S, Ramamurth K G. An extension of two machines sequencing problem. *Opsearch*, 1971, 8(1): 10–22

[2] Salvador M S. A Solution to a Special Class of Flow Shop Scheduling Problems. *Symposium of Theory of Scheduling and its Applications*, 1973: 83–91

[3] Gupta J N D, Lauff V, Werner F, Sotskov Y N. Heuristics for hybrid flow shops with controllable processing times and assignable due dates. *Computers & Operations Research*, 2002, 29(10): 1417–1439

[4] Alisantoso D, Khoo L P, Jiang P Y. An immune algorithm approach to the scheduling of a flexible PCB flow shop. *International Journal of Advanced Manufacturing Technology*, 2003, 22(11-12): 819–827

[5] Lin H T, Liao C J. A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, 2003, 86(2): 133–143

[6] Wang W, Hunsucker J L. An evaluation of the CDS heuristicin flow shops with multiple processors. *Journal of the Chinese Institute of industrial Engineers*, 2003, 20(3): 295–304

[7] Almeder C, Hartl R F. A meta heuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer. *International Journal of Production Economics*, 2013, 145(1): 88–95

[8] ChunLung Chen, ChuenLung Chen. Bottle neck-based heuristics to minimize tardy jobs in a flexible flow line with unrelated parallel machines. *International Journal of Production Research*, 2008, 46(22): 6415–6430

[9] Lahlou C. Modelling and solving a practical flexible job-shop scheduling problem with blocking constraints. *International Journal of Production Research*, 2011, 49(8): 2169–2182

[10] 王辉，鲁习文. 工件带到达时间的两阶段柔性流水作业的近似算法. 运筹学学报, 2007, 11(3): 86–94
(Wang H, Lu X W. Approximation algorithms for two-stage flexible flow shop scheduling subject to release dates. *Operations Research Transactions*, 2007, 11(3): 86–94

[11] 陈田，陈庆新. 具有两道工序的柔性同序加工车间任务投放策略. 工业工程, 2010, 13(5): 69–74
(Chen T, Chen Q X, Mao N, Liu J J. A simulation approach to job release for two-stage flexible flow shop. *Industrial Engineering Journal*, 2010, 13(5): 69–74)

[12] Li Z T, Chen Q X, Mao N, et al. Scheduling rules for two-stage flexible flow shop scheduling problem subject to tail group constraint. *International Journal of Production Economics*, 2013, 146(2): 667–678

[13] Moseley B, Dasgupta A, Kumar R. On scheduling in map-reduce and flow-shops. *ACM Symposium on Parallelism in Algorithms and Architectures*, 2011: 289–298

[14] Choi B C, Lee K. Two-stage proportionate flexible flow shop to minimize the makespan. *Journal of Combinatorial Optimization*, 2013, 25(1): 123–134

[15] 孙景昊，邓庆绪，孟亚坤. GPU 上两阶段负载调度问题的建模与近似算法. 软件学报, 2014, 25(2): 298–313
(Sun J H, Deng Q X, Meng Y K. Two-stage Workload Scheduling Problem on GPU Architectures Formulation and Approximation Algorithm. *Journal of Software*, 2014, 25(2): 298–313

[16] Drozdowski M. Scheduling multiprocessor tasks-an overview. *European Journal of Operational Research*, 1996, 94(2): 215–230

[17] Du J, Leung Y T. Complexity of scheduling parallel task systems. *SIAM J. Discrete Math.*, 2006, 2(4): 473–487

[18] Johannes B. Scheduling parallel jobs to minimize the makespan. *Journal of Scheduling*, 2006, 9(5): 433–452

[19] Amoura A K, Bampis E, Kenyon C, et al. Scheduling independent multiprocessor tasks. *Algorithmica*, 2002, 32(2): 247–261

[20] Jansen K, Porkolab L. Linear-time approximation schemes for scheduling malleable parallel tasks. *Algorithmica*, 2002, 32(3): 507–520

[21] Garey M R, Graham R L. Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 1975, 4(2): 187–200

[22] Ludwig W, Tiwari P. Scheduling malleable and non-malleable parallel tasks. ACM-SIAM Symposium

on Discrete Algorithms (SODA 1994), 1994, 167–176

[23] Oğuz C, Ercan M F, Cheng T C, et al. Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. *European Journal of Operational Research*, 2003, 149(2): 390–403

[23] Hoogeveen J A, Lenstra J K, Veltman B. Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. *European Journal of Operational Research*, 1996, 89(1): 172–175

Approximation Algorithm on Two Stage Flexible Flow-shop Scheduling with Parallel Machines

ZHANG Minghui

(*School of Software Technology, Dalian University of Technology, Dalian* 116620, *China*)

(*Software Engineering Department, Dalian Neusoft University of Information, Dalian* 116023, *China*)

HAN Xin

(*School of Software Technology, Dalian University of Technology, Dalian* 116620, *China*)

(*E-mail: neusoftzhang@163.com*)

Abstract In this paper we consider a two-stage flow-shop scheduling model which combines flexible flow shop with parallel scheduling. There is only one machine at the first stage and m identical parallel executing machines at the other. Each task needs size_i parallel machines execute simultaneously at the second stage. The objective is to minimize the makespan. This model has been proved to be NP-hard and an approximation algorithm has been proposed as well. In this paper we describe the process of the previous algorithm and point out the limitations of algorithm approximate ratio analysis. Then we propose a 3-approximation algorithm based on parallel scheduling results. Our algorithm discards the constraints of previous 3-approximation algorithm analysis. Lastly we study the two special cases with two parallel machines and three parallel machines in the second stage respectively. We propose two approximation algorithms with 2.5 and 2.67 approximate ratio correspondingly according to list scheduling rules.

Key words flexible flow shop scheduling; parallel scheduling; approximation algorithm; approximation ratio

MR(2000) Subject Classification 03f99

Chinese Library Classification O224.10