计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200600064

# 针对经典排序问题的一种新算法的近似比分析

高吉吉　岳雪蓉　陈智斌
昆明理工大学理学院　昆明 650000
(965789294@qq.com)

**摘　要**　给定 $m$ 台平行机(同型机),$n$ 个工件,寻找一种分配方案,使得把这 $n$ 个工件分配到 $m$ 台机器后,整体完工时间尽可能短,这个 NP-难问题被称为经典排序问题。如果每个工件的加工时间满足一定的条件,则有望能在多项式时间内有效地得到最优的分配方案。Yue 等对加工时间满足整除性质的经典排序问题考虑了一种新的算法,该算法总是能得到这种特殊情况的最优分配。该算法在多项式时间内能够得到最优分配,是对于一般的经典排序问题的近似算法。文章在此基础上,考虑该新算法在一般问题上的近似比。文中考虑了这个新算法的两种版本,分别得到了 3/2 和 $2-1/2^q$($q\in Z^+$)的近似比。紧例子表明,文中对算法的两个版本的分析都是最优的。

**关键词**:经典排序;近似算法;多项式时间算法;紧例子;一维装箱问题
**中图法分类号**　TP301.6

## Approximate Ratios Analysis of New Algorithm for Classical Parallel Scheduling

GAO Ji-ji, YUE Xue-rong and CHEN Zhi-bin
School of Science and Technology, Kunming University of Science and Technology, Kunming 650000, China

**Abstract**　Given $m$ parallel machines(identical machines) and $n$ jobs, we need to find a distribution scheme so that the overall completion time is as short as possible after these $n$ jobs are allocated to $m$ machines. This NP-hard problem is called classical scheduling problem. If the processing time of each job meets certain conditions, it is expected to get the optimal allocation scheme effectively. Yue et al. consider a new algorithm for the classical scheduling problem that the processing time satisfies the divisional property, which can always obtain the optimal distribution in this special case. This algorithm can obtain the optimal distribution in polynomial time and is an approximate algorithm for the general classical scheduling problem. On this basis, the approximate ratio of the algorithm in general problems is considered. There are two versions of the proposed algorithm, and the approximate ratios of 3/2 and $2-1/2^q$($q\in Z^+$) are obtained respectively. The tight examples provided in this paper show that our analysis of two versions of the algorithm is optimal.

**Keywords**　Classic scheduling, Approximation algorithm, Polynomial time algorithm, Tight example, One dimensional packing problem

## 1　引言

给定 $m$ 台平行机(同型机)$M=\{M_1,M_2,\cdots,M_m\}$ 和 $n$ 个工件 $J=\{J_1,J_2,\cdots,J_n\}$,每个工件 $J_j$($j=1,2,\cdots,n$)都有一个加工时间 $p_j$($p_j>0$),且每个工件只能在一台机器上加工一次,一旦开始加工就不能中断,必须加工完毕,将 $n$ 个工件安排到 $m$ 台机器上加工,使得最大完工时间最小,这就是经典的平行机排序问题。用 $C_i$($i=1,2,\cdots,m$)表示第 $i$ 台机器 $M_i$ 的完工时间,最大完工时间(makespan)则用 $C_{\max}=\max\limits_{1\leqslant i\leqslant m}\{C_i\}$ 表示。为了方便,采用三元组 $\alpha|\beta|\gamma$[1] 将经典排序问题描述为 $P\|C_{\max}$。

当机器数量为 2 时(即 $m=2$),$P2\|C_{\max}$ 已经是 NP-难了,对于任意的 $m$,$P\|C_{\max}$ 问题则是强 NP-难的,因此很多文献采用近似算法来求解 $P\|C_{\max}$ 问题的近似解。常用的近似算法有由 Graham 于 1966 年提出的 LS(List-Scheduling)算法[2],其按照任意次序将工件进行排序,把每个工件分配给当前负载最小的机器,近似比为 2;另一个经典的近似算法是由 Graham 于 1969 年提出的 LPT(Longest-Processing-Time-first)算法[3],其把工件按照加工时间非增排序依次分配给当前负载最小的机器,近似比为 $4/3-1/3m$,算法的时间复杂度为 $O(n\log n)$。Deuermeyer 等[4] 在 1982 年证明了 LPT 算法的最坏比例为 4/3。Kao 等[5] 在 1990 年也使用 LPT 算法求解 $P\|C_{\max}$ 问题。Csirik 等[6] 在 1992 年进一步给出了 LPT 算法的最坏比例为 $(4m-2)/(3m-1)$。Woeginger[7] 于 1997

到稿日期:2020-06-10　返修日期:2020-08-26　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(11761042,11461081)
This work was supported by the National Natural Science Foundation of China(11761042,11461081).
通信作者:陈智斌(chenzhibin311@126.com)

年对经典排序问题提出了多项式时间近似算法。1978 年，Coffman 等利用解装箱问题的 FFD 算法，给出一个 MULTIFIT 近似算法[8]，并证明了算法的近似比为 1.22，不久 Langston 等对 MULTIFIT 算法作出改进，得到 72/61 的近似比[9]，对于输入规模为多项式的算法，这是已知的最佳界。1984 年，Friesen 进一步证明了 MULTIFIT 算法的近似比可达 1.20，并给出了上界为 13/11 的紧例子[10]。Yue[11] 则证明了此算法的精确上界就是 13/11，从而基本结束了对 MULTIFIT 算法应用于解 $P\|C_{\max}$ 问题的讨论。近似算法讨论的另一个角度是算法达到 $\varepsilon$ 精度的问题，记 $P\|C_{\max}$ 问题的最优值为 $\mathrm{OPT}(I,m)$，其中 $I$ 是问题的一个实例，$m$ 为机器数。对于任给的 $\varepsilon>0$，Sahni 利用 $m$-划分给出了一个在 $O(n\cdot(\frac{\varepsilon^2}{n})^{m-1})$ 时间内获得上界为 $(1+\varepsilon)\mathrm{OPT}(I,m)$ 的算法[12]。Hochbaum 等则利用装箱问题与 $P\|C_{\max}$ 问题的联系得到了 $P\|C_{\max}$ 问题的 $\varepsilon$-近似解[13]，其计算量为 $O((\frac{n}{\varepsilon})^{\frac{1}{\varepsilon^2}}\cdot\log\frac{1}{\varepsilon})$。1991 年，Vel 等修改了此算法，达到 $(1+\varepsilon)\mathrm{OPT}(I,m)$ 的上界[14]，其计算量为 $O(n^{2P}\log\frac{1}{\varepsilon})$，其中 $P=\max_{1\leqslant j\leqslant n}\{p_j\}$。至今为止，经典平行机排序问题最快的多项式时间近似方案(PTAS)的运行时间为 $2^{O((1/\varepsilon^2)\log^3(1/\varepsilon))}+O(n\log n)$[15]。2016 年，Klein 等[16]给出了一个运行时间为 $2^{O((1/\varepsilon)\log^{O(1)}(1/\varepsilon))}+O(n\log n)$ 的有效多项式时间近似方案(EPTAS)。2018 年，Chen 等[17]说明在指数时间(ETH)[18]下，经典平行机排序问题不存在运行时间为 $2^{O(1/\varepsilon^{1-\delta})}+n^{O(1)}(\delta>0)$ 的多项式时间近似方案。在指数时间假设下，继续改进多项式时间近似方案的运行时间几乎是徒劳。本文则是对经典排序问题的一种新算法的近似比进行了分析。

对于 $P\|C_{\max}$ 问题，如果每个工件的加工时间满足一定的条件，我们有望能在多项式时间内有效地得到最优的分配方案。Yue 等[19]对加工时间满足整除性质的经典排序问题考虑了一种新的算法，我们之为算法 $\mathscr{A}$，该算法总是能在多项式时间内得到这种特殊情况的最优分配。本文此基础上考虑了算法 $\mathscr{A}$ 的两个版本。我们发现这个在多项式时间内能够得到最优分配的算法是对于一般的经典排序问题的近似算法，能够得到较好的近似比。本文分析了算法 $\mathscr{A}$ 在一般问题上的近似比，针对这两个不同的版本，证明了其近似比分别为 3/2 和 $2-1/2^q(q\in Z^+)$，并给出相应的紧例子。我们提供的紧例子表明，本文对算法 $\mathscr{A}$ 的两个版本的近似比分析都是最优的。

本文第 2 节分别给出了 $\mathscr{A}$ 算法的两个版本，并对这两个版本加以详细说明；第 3 节是对算法近似比的证明并给出了相应的紧例子；最后对全文进行总结和展望。

## 2 算法 $\mathscr{A}$ 的两个版本

在介绍算法 $\mathscr{A}$ 之前，首先介绍装箱问题。

装箱问题可描述为：给定 $n$ 个有尺寸的物体以及若干容积相同的箱子，将这些物体装入箱子中，目标是使得所用的箱子数尽可能少。不失一般性，可假定箱子的容量为 1，装箱问题用数学语言叙述如下：给定非负序列 $a_1,a_2,\cdots,a_n$，且 $a_i\in(0,1](i=1,2,\cdots,n)$，确定一个整数 $k\in N$ 和一个映射 $f:\{1,2,\cdots,n\}\to\{1,2,\cdots,k\}$，其中 $\sum_{i:f(i)=j}a_i\leqslant 1$ 对任意的 $j\in\{1,\cdots,k\}$ 成立，使得 $k$ 达到最小。

不难看出，装箱问题与 $P\|C_{\max}$ 问题是密切相关的。存在完工时间为 $t$ 的排序当且仅当能将大小为 $p_1,p_2,\cdots,p_n$ 的 $n$ 个物体装入 $m$ 个容量都为 $t$ 的箱子中[20]。基于两个问题之间的联系，对排序问题提出了算法 $\mathscr{A}$。

算法 $\mathscr{A}$ 运行的每一轮都可视为一个装箱过程。在第一轮中，将每台机器看作容量大小为 $p_1$ 的箱子，将工件 $J_j$ 安排在机器 $M_i$ 上加工看作将 $p_j$ 大小的物品放入第 $i$ 个箱子。我们把 $p_1$ 放入第一个箱子，$p_2$ 放入第二个箱子，对于 $p_j$，如果 $p_j$ 不适合(不能装入)当前的箱子，就把 $p_j$ 放入下一个箱子，以此类推。下面介绍算法的两个版本。

在无空隙版本中，算法在下一轮的装箱过程中，总是尽可能地把上一轮中未装满的箱子的空隙填补起来。在算法运行的每一轮开始，都需要先更新箱子的容量，例如，算法运行到第二轮开始时，箱子的容量为当前加工时间最大值与第一轮时箱子容量之和；运行到第三轮时，箱子的容量为当前加工时间最大值与第二轮的箱子容量之和。

为方便起见，用 OPT 表示问题 $P\|C_{\max}$ 的实例的最优值，OUT 表示算法得到的输出值(即算法中的最大完工时间 $C_{\max}$)。下面对算法中的一些符号进行说明：$M_i^{(0)}\leftarrow 0$ 表示机器的初始状态，$M_i^{(r)}$ 表示运行第 $r$ 轮时在机器 $M_i$ 上的负载；$i$ 为输入参数；$b$ 与 $d$ 是工件的编号。

**算法 1** 算法 $\mathscr{A}$ 的无空隙版本

输入：$P\|C_{max}$ 的一个实例 I(M,J,$p_j$)

输出：最大完工时间 $C_{max}$

Begin

1. 将工件按照加工时间非增进行排序编号，对工件重新索引，使得 $p_1\geqslant p_2\geqslant\cdots\geqslant p_n$.
2. $C\leftarrow 0, r\leftarrow 1, M_i^{(0)}\leftarrow 0$.
3. 若 $J\neq\emptyset$，则调用子程序将工件集 J 划分为 $J_I$ 和 $J_{II}$；否则，执行步骤 6.
4. $C\leftarrow C+\max\{p_j\mid J_j\in J_I\}$.
5. $J=J/J_I; r\leftarrow r+1$. 返回 3.
6. 输出 C(即 $C_{max}$).

子程序

1. 将工件按照加工时间非增进行排序编号，对工件重新索引，使得 $p_1\geqslant p_2\geqslant\cdots\geqslant p_n$.
2. $J_I\leftarrow\emptyset$. 将工件 $J_1$ 安排到机器 $M_1$ 上. $J_I=J_I\cup\{J_1\}$. $i\leftarrow 1, b\leftarrow i, i=i+1$.
3. while $i\leqslant m$ do

将工件 $J_{b+1},J_{b+2},\cdots,J_d$ 安排到机器 $M_i$ 上，$M_i^{(r)}\leftarrow M_i^{(r-1)}+\sum_{j=b+1}^{d}p_j$，使得

$M_i^{(r)}\leqslant C+p_1, M_i^{(r)}+p_{d+1}>C+p_1$. $J_I=J_I\cup\{J_{b+1},J_{b+2},\cdots,J_d\}$. $b\leftarrow d$. $i\leftarrow i+1$.

4. 返回 $J_I$.

End

在有空隙版本中，算法是将运行的每一轮都视作一个新的装箱过程，每一轮结束后不管箱子是否装满，都要重新开始新的一轮装箱，这就使得每一轮结束后，存在一些箱子没有装

满，这些未装满的箱子和第一个箱子相比有一个空隙。也就是说，在算法运行的任意一轮，机器 $M_2$，$M_3$，…，$M_m$ 的负载不会大于机器 $M_1$ 的负载。

**算法 2** 算法 $\mathcal{A}$ 的有空隙版本

输入：$P \| C_{max}$ 的一个实例 I(M,J,$p_j$)

输出：最大完工时间 $C_{max}$

Begin

1. 将工件按照加工时间非增进行排序编号，对工件重新索引，使得 $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_n$.
2. C←0，r←1.
3. 若 J≠Ø，则调用子程序将工件集 J 划分为 $J_I$ 和 $J_{II}$；否则，执行步骤 6.
4. C←C+max{$p_j | J_j \in J_I$}.
5. J=J/$J_I$；r←r+1. 返回 3.
6. 输出 C(即 $C_{max}$).

子程序

1. 将工件按照加工时间非增进行排序编号，对工件重新索引，使得 $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_n$.
2. $J_I$←Ø. 将工件 $J_1$ 安排到机器 $M_1$ 上. $J_I = J_I \cup \{J_1\}$. i←1，b←i，i=i+1.
3. while i≤m do

将工件 $J_{b+1}$，$J_{b+2}$，…，$J_d$ 安排到机器 $M_i$ 上，使得 $\sum_{j=b+1}^{d} p_j \leqslant p_1$，$\sum_{j=b+1}^{d+1} p_j > p_1$. $J_I = J_I \cup \{J_{b+1}, J_{b+2}, \cdots, J_d\}$. b←d. i←i+1.

4. 返回 $J_I$.

End

为了更好地理解这两种算法，下面给出一个具体例子对算法进行进一步说明。

例 1 考虑经典排序问题 $P \| C_{max}$。给定 3 台机器和 8 个工件，每个工件以及工件的加工时间如表 1 所列。分别用算法 1 和算法 2 把工件安排到机器上进行加工。

表 1 工件及工件的加工时间

Table 1 Jobs and its processing time

| 工件 | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ |
|---|---|---|---|---|---|---|---|---|
| 加工时间 | 6 | 5 | 4 | 4 | 3 | 2 | 2 | 2 |

按算法 1 对工件进行安排，排列结果如图 1 所示。

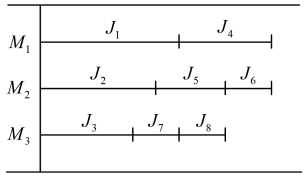


图 1 算法 1 的排序结果

Fig. 1 Scheduling result of algorithm 1

此时，OUT=OPT=10，算法的输出值与最优值相等。

按算法 2 对工件进行安排，排列结果如图 2 所示。

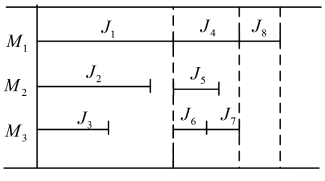


图 2 算法 2 的排序结果

Fig. 2 Scheduling result of algorithm 2

此时，OUT=12，OPT=10，$\frac{OUT}{OPT}=\frac{6}{5}$。

## 3 算法近似比的证明

### 3.1 算法 1 近似比证明

算法首先将工件按加工时间非增的顺序排列，即 $p_i \geqslant p_{i+1}$($i=1,2,\cdots,n-1$)，再依次安排工件到机器上加工。我们不妨设排在第一台机器 $M_1$ 上的工件依次为 $J_1$，$J_{k_2}$，$J_{k_3}$，…，$J_{k_{q-1}}$，$J_{k_q}$($q \in Z^+$)，由算法 1(特别地，当 $q=1$ 时，工件 $J_{k_1}$ 即工件 $J_1$，即 $p_{k_1}$ 为 $p_1$)，有 $p_1 \geqslant p_{k_2} \geqslant p_{k_3} \geqslant \cdots \geqslant p_{k_{q-1}} \geqslant p_{k_q}$。

在证明算法近似比之前，我们先引入引理 1。

**引理 1** 当 $q \geqslant 3$($q \in Z^+$)时，算法 1 基于 $P \| C_{max}$ 问题的最优值有两个下界：

(1) $p_1 + p_{k_2} + p_{k_3} + \cdots + p_{k_{q-2}} < OPT$；

(2) $p_1 + p_{k_2} + p_{k_3} + \cdots + p_{k_{q-2}} + p_{k_q} < OPT$。

证明：根据 $q$ 的取值不同，我们分两种情况讨论。

情况 1 $q=3$($p_{k_1}$ 就是 $p_1$)。此时安排在机器 $M_1$ 上加工的工件有 $J_1$，$J_{k_2}$ 和 $J_{k_3}$，算法的输出值为 $OUT = p_1 + p_{k_2} + p_{k_3}$。

(1) 工件 $J_1$ 的加工时间显然是 OPT 的一个下界，即 $p_1 < OPT$，(1)得证。

(2) 设 $J' = \{J_1, J_2, \cdots, J_{k_3-1}\}$，在安排工件 $J_{k_3}$ 之前，即当把 $J'$ 中的所有工件安排到机器上时，由算法 1，每台机器的负载均大于 $p_1$，于是就有 $mp_1 < \sum_{J_j \in J'} p_j$(否则，如果有一台机器的负载小于或等于 $p_1$，就可以把 $J_{k_3}$ 安排到这台机器上，则 $p_1 < OPT_{J'} < OPT$)，因此 $p_1 + p_{k_3} < OPT$，(2)得证。

情况 2 $q \geqslant 4$。此时，按照算法的排列可知 $OUT = p_1 + p_{k_2} + \cdots + p_{k_{q-1}} + p_{k_q}$。

(1) 设 $J'' = \{J_1, J_2, \cdots, J_{k_q-1}\}$，在安排工件 $J_{k_q}$ 之前，也就是把 $J''$ 中的所有工件安排到机器上时，由算法 1 可知，此时每台机器的负载均大于 $p_1 + p_{k_2} + \cdots + p_{k_{q-2}}$，即不等式 $m(p_1 + p_{k_2} + \cdots + p_{k_{q-2}}) < \sum_{J_j \in J''} p_j$ 成立(否则，如果有一台机器的负载小于等于 $p_1 + p_{k_2} + \cdots + p_{k_{q-2}}$，就可以把 $J_{k_q}$ 安排到这台机器上)，因此 $p_1 + p_{k_2} + \cdots + p_{k_{q-2}} < OPT$，(1)得证。

(2) 由情况 2(1)，在把 $J''$ 中的所有工件，即工件 $J_1$，$J_2$，…，$J_{k_q-1}$ 安排到机器上时，每台机器的负载均大于 $p_1 + p_{k_2} + p_{k_3} + \cdots + p_{k_{q-2}}$，且还有剩余工件 $\{J_{k_q}, \cdots, J_n\}$ 没有被加工，则 $OPT_{J'' \cup \{J_{k_q}\}} > p_1 + p_{k_2} + \cdots + p_{k_{q-2}} + p_{k_q}$，因此 $p_1 + p_{k_2} + \cdots + p_{k_{q-2}} + p_{k_q} < OPT$，(2)得证。

综上，引理 1 证毕。

**定理 1** 对于经典平行机排序问题，设按照算法 1 安排在第一台机器 $M_1$ 上的工件依次为 $J_1$，$J_{k_2}$，$J_{k_3}$，…，$J_{k_{q-1}}$，$J_{k_q}$($q \in Z^+$)。算法 1 对于 $P \| C_{max}$ 问题的近似比为 3/2。

证明：根据 $q$ 的取值不同，我们分 4 种情况讨论。

情况 1 $q=1$。此时安排在机器 $M_1$ 上的工件只有 $J_1$，$OUT = p_1$。根据算法 1，机器 $M_2$ 到机器 $M_m$ 的负载不会超过 $p_1$，则 $OUT = p_1 < \frac{3}{2}OPT$。具体如图 3 所示。

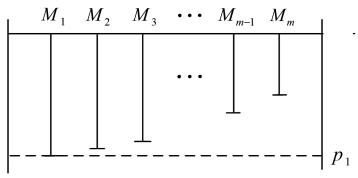


图 3 $q=1$ 的示例图

Fig. 3 Example diagram of $q=1$

情况 2 $q=2$。此时安排在机器 $M_1$ 上的工件有 $J_1$ 和 $J_{k_2}$，$\mathrm{OUT}=p_1+p_{k_2}$。根据算法 1，最终每台机器的负载均大于 $\frac{1}{2}p_1$（否则，如果有一台机器的负载小于或等于 $\frac{1}{2}p_1$，按照算法 1 工件按加工时间非增排序，有 $p_{k_2}\leqslant\frac{1}{2}p_1$，就可以把 $J_{k_2}$ 安排到这台机器上），于是 $\frac{1}{2}p_1+p_{k_2}<\mathrm{OPT}$。另外，最大的加工时间显然是 OPT 的一个下界，即 $p_1<\mathrm{OPT}$。因此，$\mathrm{OUT}=p_1+p_{k_2}=\frac{1}{2}p_1+(\frac{1}{2}p_1+p_{k_2})<\frac{1}{2}\mathrm{OPT}+\mathrm{OPT}=\frac{3}{2}\mathrm{OPT}$。具体如图 4 所示。

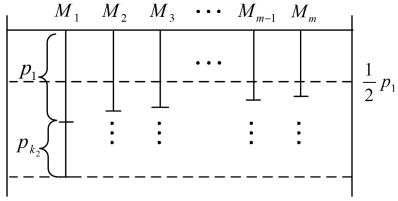


图 4 $q=2$ 的示例图

Fig. 4 Example diagram of $q=2$

情况 3 $q=3$。此时安排在机器 $M_1$ 上的工件有 $J_1$，$J_{k_2}$ 和 $J_{k_3}$，$\mathrm{OUT}=p_1+p_{k_2}+p_{k_3}$。由引理 1，OPT 有两个下界：$p_1<\mathrm{OPT}$，$p_1+p_{k_3}<\mathrm{OPT}$。考虑工件 $J_{k_2}$。

(1) $p_{k_2}\leqslant\frac{1}{2}p_1$。此时，$p_{k_2}\leqslant\frac{1}{2}p_1<\frac{1}{2}\mathrm{OPT}$，即 $\mathrm{OUT}=p_1+p_{k_2}+p_{k_3}=(p_1+p_{k_3})+p_{k_2}<\mathrm{OPT}+\frac{1}{2}\mathrm{OPT}=\frac{3}{2}\mathrm{OPT}$，具体如图 5 所示。

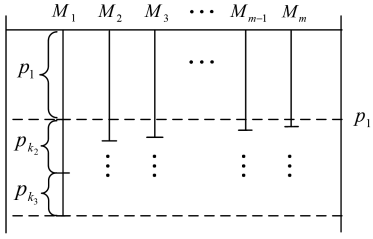


图 5 $q=3(1)$的示例图

Fig. 5 Example diagram of $q=3(1)$

(2) $p_{k_2}>\frac{1}{2}p_1$，此时 $p_2\geqslant p_3\geqslant\cdots\geqslant p_{k_2-1}\geqslant p_{k_2}>\frac{1}{2}p_1$。这说明，在安排工件 $J_{k_2}$ 之前，每台机器上只安排了一个工件，即 $k_2=m+1$，于是有 $2p_{k_2}\leqslant p_{k_2}+p_m<\mathrm{OPT}$，可得 $p_{k_2}<\frac{1}{2}\mathrm{OPT}$。因此，$\mathrm{OUT}=p_1+p_{k_2}+p_{k_3}=(p_1+p_{k_3})+p_{k_2}<\mathrm{OPT}+\frac{1}{2}\mathrm{OPT}=\frac{3}{2}\mathrm{OPT}$，具体如图 6 所示。

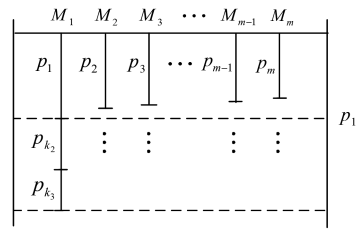


图 6 $q=3(2)$的示例图

Fig. 6 Example diagram of $q=3(2)$

情况 4 $q\geqslant4$。此时 $\mathrm{OUT}=p_1+p_{k_2}+\cdots+p_{k_{q-1}}+p_{k_q}$。由引理 1，OPT 有两个下界：$p_1+p_{k_2}+\cdots+p_{k_{q-2}}<\mathrm{OPT}$；$p_1+p_{k_2}+\cdots+p_{k_{q-2}}+p_{k_q}<\mathrm{OPT}$。

只需考虑工件 $J_{k_{q-1}}$。根据算法 1，$p_1\geqslant p_{k_2}\geqslant\cdots\geqslant p_{k_{q-2}}\geqslant p_{k_{q-1}}$，则 $p_{k_{q-1}}\leqslant\frac{1}{q-2}(p_1+p_{k_2}+\cdots+p_{k_{q-2}})\leqslant\frac{1}{q-2}\mathrm{OPT}\leqslant\frac{1}{2}\mathrm{OPT}$，因此有：

$$\begin{aligned}\mathrm{OUT}&=p_1+p_{k_2}+\cdots+p_{k_{q-2}}+p_{k_{q-1}}+p_{k_q}\\&=(p_1+p_{k_2}+\cdots+p_{k_{q-2}}+p_{k_q})+p_{k_{q-1}}<\mathrm{OPT}+\\&\quad\frac{1}{2}\mathrm{OPT}=\frac{3}{2}\mathrm{OPT}\end{aligned}$$

此情况下的示例图如图 7 所示。

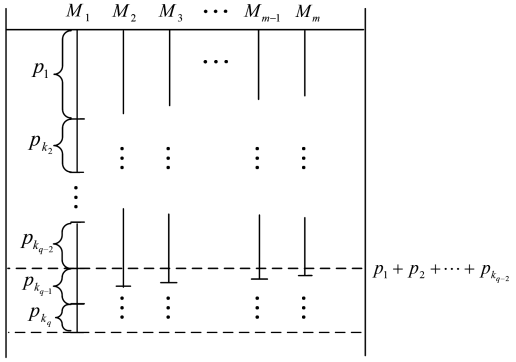


图 7 $q\geqslant4$ 的示例图

Fig. 7 Example diagram of $q\geqslant4$

综上，定理 1 证毕。

下面给出一个按算法 1 安排得到近似比为 $\frac{3}{2}$ 的紧例子。

例 2 考虑经典排序问题 $P\|C_{\max}$。有 $m$ 台机器和 $n$ 个工件，其中 $n=2m-1$，$n$ 个工件的加工时间分别为 $l$，$\frac{l}{2}+\varepsilon$，…，$\frac{l}{2}+\varepsilon$，$\frac{l}{2}$，…，$\frac{l}{2}$（$\varepsilon$ 是一个任意小的正数），其中加工时间为 $\frac{l}{2}+\varepsilon$ 和 $\frac{l}{2}$ 的工件各有 $m-1$ 个。

由算法 1 得 $\mathrm{OUT}=\frac{3}{2}l$，但 $\mathrm{OPT}=l+\varepsilon$。

### 3.2 算法 2 近似比证明

证明算法 2 的近似比之前，先对一些符号进行解释说明。$A_i^{(r)}$ 表示算法运行第 $r$ 轮时安排在机器 $M_i$ 上的工件集，$p(M_i;rth;run)=\sum_{J_j\in A_i^{(r)}}p_j$ 表示算法运行第 $r$ 轮时机器 $M_i$ 的完工时间，即被安排在机器 $M_i$ 上的工件的加工时间总和。

为方便理解，可参见图 8 的示例图。

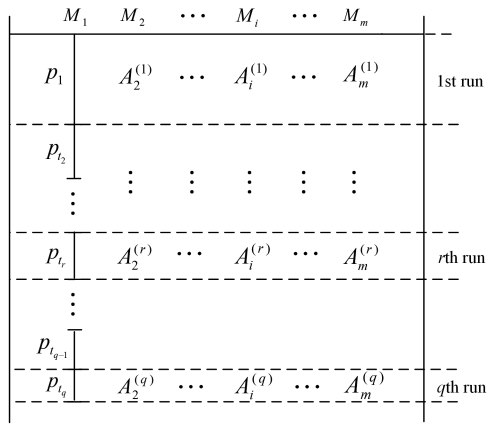


图 8 算法 2 的示例图

Fig. 8 Example diagram of algorithm 2

**定理 2** 对于经典平行机排序问题，设按算法 2 安排在第一台机器 $M_1$ 上的工件依次为 $J_1,J_{t_2},J_{t_3},\cdots,J_{t_{q-1}},J_{t_q}$（$q\in Z^+$），则算法 2 的近似比为 $2-1/2^q$（$q\in Z^+$）。

证明：由算法 2，有 $p_1\geqslant p_{t_2}\geqslant p_{t_3}\geqslant\cdots\geqslant p_{t_{q-1}}\geqslant p_{t_q}$（实际上，$p_{t_1}$ 就是 $p_1$）。$\mathrm{OUT}=p_1+p_{t_2}+\cdots+p_{t_{q-1}}+p_{t_q}$。

不难发现，在算法运行第 $r$ 轮时，机器 $M_{i+1}$ 上最大的加工时间大于机器 $M_1$ 与机器 $M_i$ 在第 $r$ 轮时的负载的差值；机器 $M_i$ 上任意工件的加工时间都不小于机器 $M_{i+1}$ 上任意工件的加工时间。在算法运行第 $r+1$ 轮时，机器 $M_1$ 上工件的加工时间大于机器 $M_1$ 与机器 $M_m$ 在第 $r$ 轮时的负载的差值；机器 $M_m$ 在第 $r$ 轮时的任意工件的加工时间都不小于机器 $M_1$ 在第 $r+1$ 轮时工件的加工时间。用公式表示为：

$$\begin{cases}\max\limits_{J_j\in A_{i+1}^{(r)}}p_j>p_{t_r}-p(M_i;r\mathrm{th};run)\\ \min\limits_{J_j\in A_i^{(r)}}p_j\geqslant\max\limits_{J_j\in A_{i+1}^{(r)}}p_j\end{cases}\tag{1}$$

$$\begin{cases}p_{t_{r+1}}>p_{t_r}-p(M_m;r\mathrm{th};run)\\ \min\limits_{J_j\in A_m^{(r)}}p_j\geqslant p_{t_{r+1}}\end{cases}\tag{2}$$

由式(1)和式(2)有：

$$p(M_i;r\mathrm{th};run)>\frac{1}{2}p_{t_r},r=1,\cdots,q-1,i=2,\cdots,m$$

也就是说，在算法运行的任意一轮，第 2 台机器到第 $m$ 台机器的负载都严格大于第 1 台机器负载的一半。因此，在第 $q-1$ 轮结束后，所有机器的负载都大于 $\frac{1}{2}(p_1+p_{t_2}+\cdots+p_{t_{q-1}})$。不妨设负载最小的机器为 $M_j$，则 $\sum\limits_{J_j\in M_j}p_j>\frac{1}{2}(p_1+p_{t_2}+\cdots+p_{t_{q-1}})$。

此时，工件 $J_{t_q},\cdots,J_n$ 还没有被加工。于是有：

$$\sum_{J_j\in M_j}p_j+p_{t_q}\leqslant\mathrm{OPT}$$

$$\frac{1}{2}(p_1+p_{t_2}+\cdots+p_{t_{q-1}})+p_{t_q}\leqslant\mathrm{OPT}$$

下面用数学归纳法证明定理 2。

(1)当 $q=1$ 时，安排在机器 $M_1$ 上的工件只有 $J_1$，$\mathrm{OUT}=p_1=\mathrm{OPT}$。定理 2 成立；

(2)当 $q=k-1$ 时，假设定理 2 成立，则有 $\mathrm{OUT}=p_1+p_{t_2}+\cdots+p_{t_{k-1}}\leqslant(2-\frac{1}{2^{k-1}})\mathrm{OPT}$；

(3)当 $q=k$ 时，有：

$$\begin{aligned}\mathrm{OUT}&=p_1+p_{t_2}+\cdots+p_{t_{k-1}}+p_{t_k}\\&=\frac{1}{2}(p_1+p_{t_2}+\cdots+p_{t_{k-1}})+\frac{1}{2}(p_1+p_{t_2}+\cdots+p_{t_{k-1}})+p_{t_k}\\&\leqslant\frac{1}{2}(2-\frac{1}{2^{k-1}})\mathrm{OPT}+\mathrm{OPT}\\&=(2-\frac{1}{2^k})\mathrm{OPT}\end{aligned}$$

定理 2 成立。

综上，定理 2 证毕。

下面给出一个按算法 2 安排得到近似比为 $2-1/2^q$（$q\in Z^+$）的紧例子。

例 3 考虑经典排序问题 $P\parallel C_{\max}$。有 $m$ 台机器和 $n$ 个工件，其中 $m=2^q$，$n=qm+1$（$q$ 是一个正整数）。$n$ 个工件的加工时间分别为 $l,\frac{l}{2}+\varepsilon,\cdots,\frac{l}{2}+\varepsilon,\frac{l}{2},\cdots,\frac{l}{4}+\varepsilon,\frac{l}{4},\cdots,\frac{l}{2^q}+\varepsilon,\cdots,\frac{l}{2^q}+\varepsilon$（$\varepsilon$ 是一个任意小的正数），这里加工时间为 $\frac{l}{2}+\varepsilon,\frac{l}{4}+\varepsilon,\cdots,\frac{l}{2^q}+\varepsilon$ 的工件各有 $m-1$ 个。

按算法 2 把工件安排到机器上加工，结果如图 9 所示。

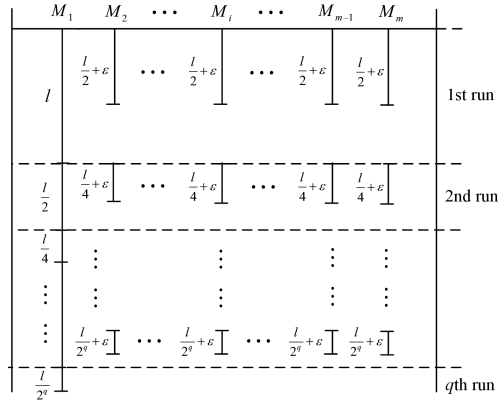


图 9 算法 2 对例 3 排序的结果

Fig. 9 Result of scheduling example 3 according to algorithm 2

此时，$\mathrm{OUT}=l+\frac{l}{2}+\frac{l}{4}+\cdots+\frac{l}{2^q}=(2-\frac{1}{2^q})l$，但 $\mathrm{OPT}=l+q\varepsilon$（$q\in Z^+$，$\varepsilon$ 是一个任意小的正数）。

**结束语** 经典排序问题已被证明是 NP-难的，但如果每个工件的加工时间满足一定的条件，经典排序问题有望能得到最优的分配方案。本文在 Yue 等提出的算法的基础上，考虑两种不同的版本，并给出了算法的两个版本的近似比分别为 3/2 和 $2-1/2^q$（$q\in Z^+$），然后分析了这一算法在一般问题上的复杂性。我们提供的紧例子表明，我们对算法的两个版本的分析都是最优的。

从计算复杂性的角度来说，现实中很多组合优化问题都是 NP-难的。对于这些 NP-难问题，如何设计近似算法对一般的 NP-难问题进行有效地求解，或限制一些条件，使得问题在多项时间内可解；研究一些特殊条件的多项式时间算法是否为解决一般 NP-难问题的近似算法；寻找使得问题可解的这些条件以及研究这些条件下的多项式时间算法或多项式时间近似方案都是我们今后研究的方向。此外，装箱的思想可

推动平行机排序问题的进一步研究，希望本文工作对正在研究此问题的研究工作者有所启发，进而提出一些新的有效算法来对问题给出合理的解答，从而更好地解决平行机排序问题。

## 参 考 文 献


[1] GRAHAM R L, LAWLER E L, LENSTRA J K, et al. Optimization and Approximation in Determini-stic Sequencing and Scheduling: A Survey[J]. Annals of Discrete Mathematics, 1979, 5(1): 287-326.

[2] GRAHAM R L. Bounds for Certain Multiprocessing Anomalies [J]. Bell System Technical Journal, 1966, 45(9): 1563-1581.

[3] GRAHAM R. L. Bounds on multiprocessing time anomalies[J]. SIAM Journal on Applied Mathematics, 1969, 17(2): 416-429.

[4] DEUERMEYER B L, FRIESEN D K, LANGSTON M A. Scheduling to Maximize the Minimum Processor Finish Time in a Multi-processor System[J]. Siam Journal on Algebraic Discrete Methods, 1982, 3(2): 190-196.

[5] KAO T Y, ELSAYED E A. Performance of the LPT algorithm in multiprocessor scheduleing[J]. Computers & Operations Research, 1990, 17(4): 365-373.

[6] CSIRIK J, KELLERER H, WOEGINGER G. The exact LPT-bound for maximizing the minimum completion time[J]. Operations Research Letters, 1992, 11(5): 281-287.

[7] WOEGINGER G J. A polynomial-time approximation scheme for maximizing the minimum machine completion time [J]. Operations Research Letters, 1997, 20(4): 149-154.

[8] COFFMAN E G, GAREY M R, JOHNSON D S. An Application of Bin-Packing to Multiprocessor Scheduling[J]. Siam Journal on Computing, 19787(1): 1-17.

[9] LANGSTON M A. Processor scheduling with improved heuristic algorithms[M]. Texas A&M University, 1981.

[10] FRIESEN D K. Tighter Bounds for the Multifit Processor Scheduling Algorithm[J]. Siam Journal on Computing, 1984, 13(1): 170-181.

[11] YUE M. On the exact upper bound for the multifit processor scheduling algorithm[J]. Annals of Operations Research, 1990, 24(1): 233-259.

[12] SAHN I, SARTAJ K. Algorithms for Scheduling Independent Tasks[J]. Journal of the ACM, 1976, 23(1): 116-127.

[13] HOCHBAUM D S, SHMOYS D B. Using dual approximation algorithms for scheduling problems theoretical and practical results[J]. Journal of the ACM(JACM), 1987, 34(1): 144-162.

[14] VEL H V D, SHIJIE S. A modification of Hochbaum and Shmoys' algorithm for scheduling problems[J]. BIT Numerical Mathematics, 1991, 31(1): 50-52.

[15] JANSEN K. An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables[J]. SIAM Journal on Discrete Mathematics, 2010, 24: 457-485.

[16] KLEIN K M, JANSEN K, VERSCHAE J. Closing the gap for makespan scheduling via sparsification techniques[C]// Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming. 2016, 55(72): 1-13.

[17] CHEN L, JANSEN K, ZHANG G. On the optimality of exact and approximation algorithms for scheduling problems[J]. Journal of Computer and System Sciences, 2018, 96(2): 89-99.

[18] IMPAGLIAZZO R, PATURI R, ZANE F. Which problems have strongly exponential complexity? [J]. Journal of Computer and System Science, 2001, 63: 512-530.

[19] YUE X R, GAO J J, CHEN Z B. A polynomial time algorithm for scheduling on processing time constraints[C]// ACM International Conference Proceeding Series, 2019 the 9th International Conference on Communication and Network Security. 2019: 109-113.

[20] 近似算法[M]. 郭效江，方志奇，农庆琴，译. 北京：北京高等教育出版社，2010：74-75.


**GAO Ji-ji**, born in 1995, postgraduate. His main research interests include combinatorial optimization and so on.

**CHEN Zhi-bin**, born in 1979, Ph.D, associate professor. His main research interests include combinatorial optimization, graph theory, approximation algorithms and operations research.