浙江理工大学学报，2022，47(5)：791-797
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673-3851(n).2022.05.020

工件加工需要额外资源的平行机调度问题

胡觉亮，李志林，董建明
（浙江理工大学理学院，杭州 310018）

摘　要：针对工件加工需要额外资源且额外资源有两个单位可用的平行机调度问题，目标是极小化最大完工时间，给出了一个近似算法。首先，说明了该问题是NP－难的，并给出了该问题的整数规划模型和最优解的下界；然后，设计了该问题的一个近似算法，通过分析算法的性质，证明了该算法的最坏情况界不大于 $3-2/m$；最后，通过大量的数值实验验证算法在平均情况下的性能。数值实验结果表明，该算法在工件数、额外资源数和机器数等因素扰动下都体现了较好的性能。该算法可为需要额外资源调度问题的高效算法设计提供参考。

关键词：额外资源；平行机；调度问题；整数规划；近似算法
中图分类号：O223　　**文献标志码**：A　　**文章编号**：1673-3851 (2022) 09-0791-07

The parallel machine scheduling problem for job processing requiring extra resources

HU Jueliang, *LI Zhilin*, *DONG Jianming*
(School of Science, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: An approximate algorithm is proposed to minimize the maximum completion time for parallel machine scheduling problems where the job processing requires extra resources and two units of extra resources are available. Firstly, the problem is shown to be NP-hard, and the integer programming model of the problem and the lower bound of the optimal solution are given. Then, an approximate algorithm of the problem is designed. By analyzing the properties of the algorithm, it is proved that the worst-case bound of the algorithm is not greater than $3-2/m$. Finally, a large number of numerical experiments are carried out to verify the performance of the algorithm under average conditions. The numerical results show that the algorithm is good under the disturbance of the number of jobs, the number of extra resources and the number of machines. This algorithm can provide a reference for efficient algorithm design with resource scheduling problem.

Key words: additional resources; parallel machine; scheduling problem; integer programming; approximate algorithm

0　引　言

调度问题[1-3]是运筹学与组合优化领域的一个重要分支，研究在有限的资源约束情况下给出所要安排任务的一个调度，使得给定的目标函数达到最优。通常把要加工处理的任务称为工件(Job)，把工件加工所需要的基础资源或设备称为机器(Machine)，即如何安排工件在机器上的加工顺序，

收稿日期：2022－05－14　　网络出版日期：2022－06－25
基金项目：国家自然科学基金项目(11971435)
作者简介：胡觉亮(1958－　)，男，浙江杭州人，教授，主要从事运筹学与组合优化、近似算法设计及应用方面的研究。
通信作者：董建明，E-mail:djm226@163.com

使得目标函数达到最优。在现实环境中,工件加工所需要的资源除了机器以外,经常还需要其他额外的资源,例如人工或者其他辅助设备等。本文将机器以外的其他资源称为额外资源。额外资源对调度的影响是多样化的,例如:额外资源使用的数量往往可以决定工件的加工速度;额外资源有时需要不断损耗,有时又可以回收重复利用等。对于工件加工需要额外资源的调度问题,如平行机环境下的问题,其基本模型的三参数可以描述为 $P\mid \mathrm{res}\lambda\sigma\delta\mid C_{\max}$,其中:$P$ 表示平行机;$\mathrm{res}\lambda\sigma\delta$ 表示一共有 $\lambda\geqslant 1$ 种资源,在任何时刻每种资源有 σ 个单位可使用,工件加工时一次需要消耗 δ 单位资源;目标函数为极小化最后完工工件的完工时间 $C_{\max}$。

对于工件加工需要额外资源的调度问题,目前研究主要集中在只有一种额外资源且每种额外资源只有一个单位可用的问题。对于只有一种额外资源且机器环境为专用机(即工件加工需要在指定机器上进行加工,用"PD"表示)的调度问题 $\mathrm{PD}_3\mid \mathrm{res}111\mid C_{\max}$、$\mathrm{PD}_4\mid \mathrm{res}111\mid C_{\max}$ 和 $\mathrm{PD}_m\mid \mathrm{res}111\mid C_{\max}$,Kellerer 等[4]分别给出了多项式时间近似算法,并对问题 $\mathrm{PD}_m\mid \mathrm{res}111\mid C_{\max}$ 给出了一个多项式时间近似方案(Polynomial time approximation scheme,PTAS)。Li 等[5]从启发式算法角度,利用遗传算法对问题 $\mathrm{PD}\mid \mathrm{res}1^{\cdots}\mid C_{\max}$ 进行了求解研究。对于额外资源种类为 λ 种、机器环境为专用机的情况下的问题 $\mathrm{PD}_m\mid \mathrm{res}\lambda 11\mid C_{\max}$,Kellerer 等[6]在文献[4]的基础上,给出了一个多项式时间近似方案。对机器环境为 m 台平行机的问题 $\mathrm{P}_m\mid \mathrm{res}\lambda 11\mid C_{\max}$,Hebrard 等[7]对其在线情况和离线情况分别给出了近似算法。Godet 等[8]通过约束规划模型该问题进行了分析。Strusevich[9]对机器数为 2 的问题 $\mathrm{P}_2\mid \mathrm{res}\lambda 11\mid C_{\max}$ 给出了一个近似比为 6/5 的近似算法。对于目标函数为极小化所有工件完工时间之和的问题 $\mathrm{P}_m\mid \mathrm{res}\lambda 11\mid \sum C_j$,Janssen 等[10]给出了一个近似比为 $2-1/m$ 的近似算法。

以上结果都假设额外资源使用完后可全部回收并供后续的工件使用;还有一类额外资源可部分回收利用的调度问题是额外资源使用后可以回收一部分继续使用,当额外资源不足时工件将不能继续加工。后一类问题最早由 Kaplan 等[11]用于房屋搬迁问题研究中。Lin 等[12]首先系统研究了该类问题,给出了该问题在目标函数为极小化最后完工工件的完工时间情况下的 NP 困难性证明。Kononov 等[13]证明了该类问题在目标函数为总加权完工时间情况下是强 NP−难的。Cheng 等[14]证明了该类问题在工件加工时间是常数、工件加工可中断、资源的回收时间是常数、工件所需的资源量是常数、资源的回收量是常数等特殊情况下的模型也都是强 NP−难的。Lin[15]研究了当工件加工和资源回收都在机器上进行时的问题模型,给出了该问题的整数规划模型和动态规划算法。

上述研究都集中在每种额外资源都只有一个单位可用的情况,本文研究每种额外资源同一时刻具有两个单位可用情况下的平行机调度问题 $\mathrm{P}_m\mid \mathrm{res}\lambda 21\mid C_{\max}$,即:工件需要在 m 台平行机上进行加工,有 λ 种资源且每种资源有两个单位可用,同时工件的加工要占用一个单位的某种资源才能进行,目标是极小化最后完工工件的完工时间。对于该问题,当 $m=2$ 时,同一时刻只要机器空闲任意工件都可以选取两台机器中的一台进行加工,所以该问题等价于 $\mathrm{P}_2\parallel C_{\max}$,由于 $\mathrm{P}_2\parallel C_{\max}$ 是 NP−难的,因此问题 $\mathrm{P}_2\mid \mathrm{res}\lambda 21\mid C_{\max}$ 也是 NP−难的。对于问题 $\mathrm{P}_m\mid \mathrm{res}\lambda 21\mid C_{\max}$,容易发现:在每种资源所对应的工件的个数不大于 2 的特殊情况下,该问题等价于问题 $\mathrm{P}_m\parallel C_{\max}$。由于问题 $\mathrm{P}_m\parallel C_{\max}$ 是 NP−难的,因此问题 $\mathrm{P}_m\mid \mathrm{res}\lambda 21\mid C_{\max}$ 至少也是 NP−难的。

本文所研究的问题源于地球观测卫星数据下载调度等领域[7],且之前的研究都集中在每种额外资源都只有一个单位可用的情况,额外资源可用数大于等于两个情况下的问题,尚未见相应的近似算法结果。本文说明了该问题是 NP−难的,给出该问题的一个整数规划模型和问题最优解的下界性质;然后给出该问题的一个最坏情况界不大于 $3-2/m$ 的近似算法,并通过分析算法性质给出了算法的最坏情况界证明;最后,通过数值实验验证本文设计的近似算法在平均情况下的性能。

1 问题的描述和最优解下界

本节对问题 $\mathrm{P}_m\mid \mathrm{res}\lambda 21\mid C_{\max}$ 进行形式化描述,并给出问题的整数规划模型和最优解下界。

1.1 问题的形式化描述

本文中符号的定义如下:m,平行机的数量;M_i,第 i 台加工的机器,$i=1,2,\cdots,m$;n,需要进行加工的工件数量;N,需要加工的工件集合,$N=(J_1,J_2,\cdots,J_n)$;J_j,代表第 j 个工件,$j=1,2,\cdots,n$;p_j,工件 j 的加工时间,$j=1,2,\cdots,n$;λ,可用资源种类数;$p(N)$,所有工件的加工时间之和;$p(N^u)$,资

源 u 对应的所有工件加工时间之和,$u=1,2,\cdots,\lambda$;C_i,第 i 台机器的完工时间;$C_{\max}$,所有机器中的最大完工时间 $C_{\max}=\max(C_i)$,$i=1,2,\cdots,m$。

问题 $P_m\mid res\lambda 21\mid C_{\max}$ 的形式化描述为:有 n 个工件的集合 $N=(J_1,J_2,\cdots,J_n)$ 需要在 m 台确定的平行机上选择一台机器 $M_i(i=1,2,\cdots,m)$ 进行加工,每个工件 $J_j(j=1,2,\cdots,n)$ 的加工需要加工时间 p_j 和消耗一个单位的某种特定的额外资源。额外资源的种类数有 λ 种,且每种额外资源同一时间有两个单位可用。将所有工件按照所需不同的额外资源可以分为若干个子集,记为 $N=\{N^1,N^2,\cdots,N^\lambda\}$,其中 $N^u(u=1,2,\cdots,\lambda)$,代表工件集合 N 中所需额外资源为 u 的工件集合。目标是得到一个工件的可行调度使得所有机器中的最大完工时间 $C_{\max}$ 最小,即最后完工工件的完工时间最小。同时不妨假设机器数与资源种类数的关系满足 $m\leqslant 2\lambda$,否则机器是多余的。

1.2　问题的整数规划模型

问题 $P_m\mid res\lambda 21\mid C_{\max}$ 的整数规划模型如下:

$$\min C_{\max} \tag{1}$$

st.

$$\sum_{i=1}^{m} x_{ij}=1,j=1,2,\cdots,n \tag{2}$$

$$\sum_{j=1}^{n} z_{tij}\leqslant 1,t=1,2,\cdots,p(N) \tag{3}$$

$$C_i\geqslant t\cdot y_{tj},i=1,2,\cdots,m,j=1,2,\cdots,n \tag{4}$$

$$C_{\max}\geqslant C_i,i=1,2,\cdots,m \tag{5}$$

$$\sum_{j=1}^{n} y_{tj}\cdot b_{ju}\leqslant 2,t=1,2,\cdots,p(N),u=1,2,\cdots,\lambda \tag{6}$$

$$s_j\leqslant t\cdot y_{tj}+M(1-y_{tj}),t=1,2,\cdots,p(N),\ j=1,2,\cdots,n \tag{7}$$

$$e_j\geqslant t\cdot y_{tj},t=1,2,\cdots,p(N),j=1,2,\cdots,n \tag{8}$$

$$e_j-s_j=p_j,j=1,2,\cdots,n \tag{9}$$

$$\sum_{t=1}^{p(N)} y_{tj}=p_j,j=1,2,\cdots,n \tag{10}$$

其中:s_j 表示工件 j 的开工时间;e_j 为工件 j 的完工时间;如果工件 j 的资源若为 u,则 $b_{ju}=1$,否则 $b_{ju}=0$;如果工件 j 在机器 i 上加工,则 $x_{ij}=1$,否则 $x_{ij}=0$;在时间段 $(t-1,t]$ 内,若加工工件为 j,则 $y_{tj}=1$,否则 $y_{tj}=0$;在时间段 $(t-1,t]$ 内,若机器 i 加工工件为 j,为 $z_{tij}=1$,否则为 $z_{tij}=0$。

在该模型的约束条件中,式(1)表示所求目标函数为最小化最大完工时间;式(2)表示每个工件只能在一台机器上加工;式(3)表示每台机器上的每个时间段 $(t-1,t]$ 最多只能加工 1 个工件;式(4)和式(5)表示最大完工时间大于等于每台机器的完工时间;式(6)表示每个时段 $(t-1,t]$ 最多同时加工 2 个同种资源的工件;式(7)—(10)表示每个工件实际加工时间等于工件的大小,其中 M 为一个较大的惩罚数值,以确保所有工件的开工时间为其实际开工时间。

1.3　问题的最优解下界

令 $p(N^u)$ 表示集合 N^u 中所有工件的加工时间之和。对于问题 $P_m\mid res\lambda 21\mid C_{\max}$ 实例的任意一个可行调度 Π,令 $C_i(\Pi)$ 表示机器 M_i 的完工时间,即在机器 M_i 上最后完工工件的完工时间,显然调度 Π 的目标函数值可以表示为:

$$C_{\max}(\Pi)=\max\{C_i(\Pi)\mid i=1,2,\cdots m\},$$

且满足 $C_{\max}(\Pi)\geqslant\dfrac{p(N)}{m}$,所以对于问题实例的最优目标函数值 $C^*_{\max}$ 同样满足 $C^*_{\max}\geqslant\dfrac{p(N)}{m}$。

对于额外资源而言,由于每种额外资源只有两个单位可用,同一时刻最多有两台机器上的工件会使用同一种额外资源,则对于 $\max\limits_{1\leqslant u\leqslant\lambda} p(N^u)$ 有:

$$C_{\max}(\Pi)\geqslant\frac{1}{2}\max_{1\leqslant u\leqslant\lambda} p(N^u),$$

所以,对于问题实例的最优目标函数值 $C^*_{\max}$ 同样满足:

$$C^*_{\max}\geqslant\frac{1}{2}\max_{1\leqslant u\leqslant\lambda} p(N^u),$$

综上可得,问题 $P_m\mid res\lambda 21\mid C_{\max}$ 最优解的一个下界 C_B,即最优目标函数 $C^*_{\max}$ 满足:

$$C^*_{\max}\geqslant C_B=\max\left\{\frac{p(N)}{m},\frac{1}{2}\max_{1\leqslant u\leqslant\lambda} p(N^u)\right\} \tag{11}$$

2　近似算法设计

本节给出问题 $P_m\mid res\lambda 21\mid C_{\max}$ 的一个最坏情况界不大于 $3-2/m$ 的近似算法。算法的基本思想是贪婪算法思想,即依次将工件插入到当前部分调度中,每次考虑当前工件 J_j 的加工过程中,如果当前同时有两台机器最后加工的工件与工件 J_j 所需资源相同,则选择在这两台机器中最早完工的机器上加工;否则,将工件 J_j 放到当前所有机器中最早完工的机器上进行加工。将该算法记为算法 A,下

面给出算法的具体描述。

2.1 算法描述

算法 A 的步骤如下：

输入：待加工工件集合 $N=\{N^1,N^2,\cdots,N^\lambda\}$ 及工件的任意初始序记为 $J_1,J_2,\cdots,J_n$。

输出：可行调度 S。

步骤 1：按初始顺序取出工件 $J_j(j=1,2,\cdots,n)$，考虑工件 J_j 的加工。

步骤 1.1：若当前已安排加工工件的部分调度中同时存在两台机器 M_i、$M_k(i\neq k)$，其最后完工的工件所需额外资源都与当前工件 J_j 所需资源相同，则将工件 J_j 安排在这两台机器中完工较早的机器上进行加工。

步骤 1.2：否则，将工件 J_j 安排在当前所有机器中最早完工的机器上进行加工。

步骤 2：重复步骤 1，直到所有工件都被安排加工，输出此时得到的调度，记为 S。

2.2 算法 A 的最坏情况界证明

本节证明算法 A 得到的调度是可行的，且算法的最坏情况界不大于 $3-2/m$。首先给出如下算法解的性质，其中 $e_{\min}$ 为所有机器中最早完工机器对应的完工时间。

引理 1 算法 A 所得的调度 S 是可行的，且所有机器一旦开始加工后就不会产生空闲，直到机器完工为止。对于每台机器 M_i，在时刻 $e_{\min}$ 后，如该机器尚未完工则该机器上加工工件所需额外资源是唯一的。

证明 首先，算法所得调度 S 的可行性是显然的，因为算法在安排每个工件的加工时已经考虑工件所需额外资源的数量限制，即每个工件的安排都是可行的，则最终所得调度也是可行的。下面证明机器无空闲及时刻 $e_{\min}$ 后加工工件所需额外资源唯一。

用数学归纳法，工件数 $n=1$ 时显然成立，假设对 n 个工件结论依然成立。接下来讨论安排第 $n+1$ 个工件时的情况。

根据假设，当前机器上已经安排前 n 个工件的加工，机器上也无额外空闲时间，且时刻 $e_{\min}$ 之后机器上所加工工件所需资源是唯一的，则根据算法 A 安排第 $n+1$ 个工件时，有如下两种情况：

a)若存在两台机器 M_i 和 $M_k(i\neq k)$，这两台机器最后完工的工件所需额外资源与 J_{n+1} 所需额外资源相同，则根据算法 A 的步骤 1.1，工件 J_{n+1} 将会被安排在这两台机器中最早完工的机器上进行加工。所以，所有机器都没有引入新的空闲，且 $e_{\min}$ 之后加工的工件只增加了 J_{n+1} 且所需额外资源种类也未增加。所以结论依然成立。

b)否则，根据算法 A 的步骤 1.2，工件 J_{n+1} 被排到当前最早完工的机器上进行加工，即在 $e_{\min}$ 发生的机器上 $e_{\min}$ 时刻之后进行加工。显然，此时最早完工机器可能会发生改变，不妨记为 M_t，且新的最早完工机器的完工时间记为 $e'_{\min}\geqslant e_{\min}$。若 M_t 为原最早完工机器，则显然 $e'_{\min}$ 时刻后其他机器上并未有新的工件加工，所以结论依然成立；若 M_t 为新的机器，则除 M_t 外其他机器上并无新的工件被加工，所以 $e'_{\min}$ 时刻后其他机器上无新的工件被加工，结论依然成立。

证毕。

引理 2 对于算法 A 所得调度 S，目标函数值满足 $C_{\max}(S)\leqslant\frac{P(N)}{m}+\left(1-\frac{1}{m}\right)\max\limits_{1\leqslant u\leqslant\lambda}p(N^u)$。

证明 不失一般性，不妨假设最大完工时间 $C_{\max}(S)$ 所对应机器为 M_m，因为除 M_m 外，其他 $(m-1)$ 台机器的完工时间 $C_i(i=1,2,\cdots,m-1)$，均满足 $C_i\geqslant e_{\min}$，则有：

$$C_{\max}(S)+(m-1)e_{\min}\leqslant C_{\max}(S)+\sum_{i\neq k}C_i=P(N) \tag{12}$$

不妨假设 M_m 上最后完工的工件所属的集合为 N^v，由引理 1 可得，$e_{\min}$ 时刻之后任意机器上加工工件所需的额外资源都是唯一的，则 $P(N^v)\geqslant C_{\max}(S)-e_{\min}$。又因为 $\max\limits_{1\leqslant u\leqslant\lambda}p(N^u)\geqslant P(N^v)$，所以可得

$$\max_{1\leqslant u\leqslant\lambda}p(N^u)\geqslant C_{\max}(S)-e_{\min} \tag{13}$$

联立式(12)和式(13)可得：

$$C_{\max}(S)\leqslant\frac{P(N)}{m}+\left(1-\frac{1}{m}\right)\max_{1\leqslant u\leqslant\lambda}p(N^u)。$$

证毕。

定理 1 对于问题 $P_m\mid res\lambda 21\mid C_{\max}$，算法 A 的最坏情况界不大于 $3-2/m$。

证明 由于问题最优解的下界性质式(11)可得：

$$C^*_{\max}\geqslant\frac{p(N)}{m}且\ 2C^*_{\max}\geqslant\max_{1\leqslant u\leqslant\lambda}p(N^u),$$

则由引理 2 可得算法 A 所得排序的目标函数值满足：

$$C_{\max}(S)\leqslant\frac{p(N)}{m}+\left(1-\frac{1}{m}\right)\max_{1\leqslant u\leqslant\lambda}p(N^u)\leqslant C^*_{\max}+2\left(1-\frac{1}{m}\right)C^*_{\max}=\left(3-\frac{2}{m}\right)C^*_{\max}。$$

由此可以得到算法解于最优解的比值为：

$$\frac{C_{\max}(S)}{C^{*}_{\max}} \leqslant 3-\frac{2}{m}。$$

证毕。

3　数值实验

为了评价算法 A 的性能,本文采用随机方式产生大量问题实例,通过数值实验验证算法在平均情况下的性能。算法采用了 Python3.10.1 来实现,电脑的运行环境为 Intel(R) Core(TM) i5-8250U CPU@1.60 GiHz、1.80 GiHz 和 8 GiB Ram。

对于小规模的实例,运用 Lingo 软件求解问题的整数规划模型,得到其最优解 $C^{*}_{\max}$,将算法 A 所得的算法解 $C^{A}_{\max}$ 与其进行比较分析;对于规模较大的实例,由于 Lingo 无法在可接受时间内得到最优解,所以本文将算法解 $C^{A}_{\max}$ 与最优解的下界 C_B 进行比较。具体比较方法采用算法的近似比,即针对实例 π 的算法解所得的目标函数值与实例 π 的最优目标函数值或最优目标函数值的下界 C_B 进行比较,得到近似比 $\alpha(\pi)=\dfrac{C^{A}_{\max}(\pi)}{C^{*}_{\max}(\pi)}$ 或 $\alpha(\pi)=\dfrac{C^{A}_{\max}(\pi)}{C_B(\pi)}$(后文用 α 表示)。随机产生的问题实例将分别从工件的加工时间、资源种类数、工件数以及机器数等因素进行组合分类。

3.1　小规模实例的数值实验

首先以工件数较少的小规模实例为例验证算法的可行性和性能。随机产生 3 个实例,其中工件数 $n=10$,资源种类数 $\lambda=3$,机器数 $m=3$,工件加工时间为[0,10]的一个均匀分布的随机数。运用 Lingo 软件求解问题的整数规划模型得到实例的最优解 $C^{*}_{\max}$,然后将算法 A 应用于实例得到算法解 $C^{A}_{\max}$。具体实例数据和实验结果如表 1 和表 2 所示。

表 1　3 个不同实例的工件加工时间和资源类型

工件	实例 1		实例 2		实例 3	
	加工时间	资源类型	加工时间	资源类型	加工时间	资源类型
J_1	8	1	6	2	14	3
J_2	4	2	11	2	2	2
J_3	7	3	12	2	13	1
J_4	15	2	7	3	13	3
J_5	10	3	1	1	13	1
J_6	15	1	3	3	9	1
J_7	1	1	4	2	1	2
J_8	13	3	12	1	5	2
J_9	6	1	8	3	4	1
J_{10}	1	1	8	1	9	2

表 2　实例的数值实验结果

实例	$C^{*}_{\max}$	$C^{A}_{\max}$	α
1	27	31	1.148
2	24	26	1.083
3	28	30	1.071

图 1—图 7 分别给出了 3 个实例最优解和算法解的工件加工示意图,从图中可以看到,算法 A 并不会导致资源冲突,并且能够产生一个可行调度。由表 2 可知,对于 3 个实例,近似比 α 都不大于 1.2。

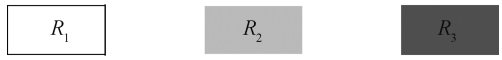

图 1　工件加工的资源种类

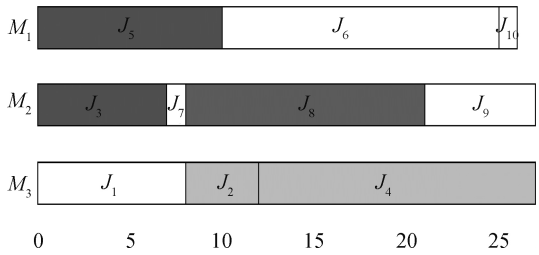

图 2　实例 1 最优解

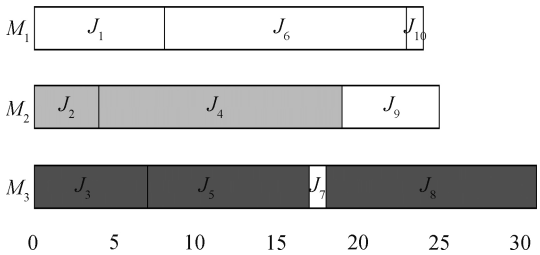

图 3　实例 1 算法解

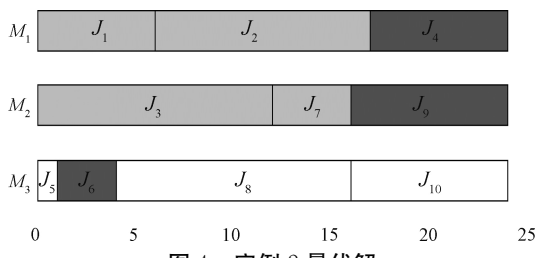

图 4　实例 2 最优解

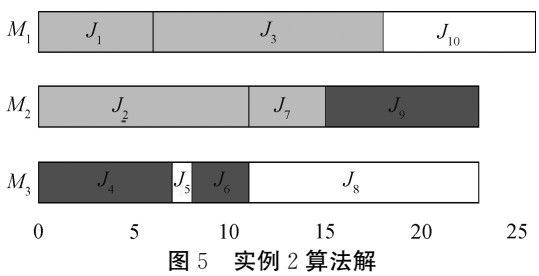

图 5　实例 2 算法解

为了进一步说明算法对小规模实例的效果,在上述其他条件不变的情况下,本文又对工件数分别为 12、15、20、25、30,每种情况随机产生 500 个实例进行分析,记 α_{avg} 为 500 个实例的近似比 α 的平均

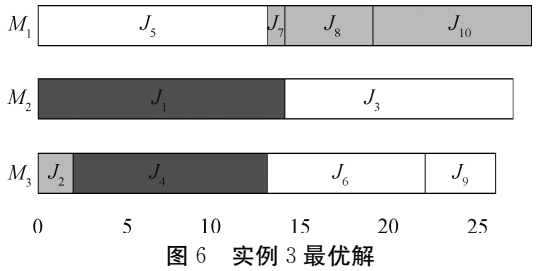

图 6 实例 3 最优解

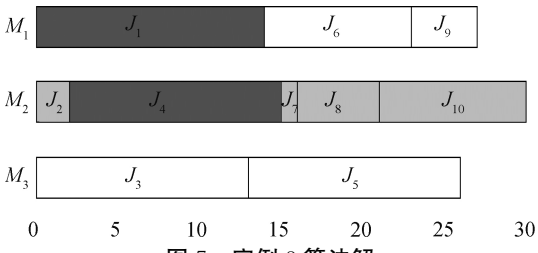

图 7 实例 3 算法解

值,记 $\alpha_{\min}$ 和 $\alpha_{\max}$ 为该类实例中最好和最坏情况下的近似比 α 值。具体结果如表 3 所示。

表 3 小规模实例的实验结果

工件个数	$\alpha_{\min}$	$\alpha_{\max}$	α_{avg}
$n=12$	1.000	1.461	1.129
$n=15$	1.000	1.278	1.104
$n=20$	1.000	1.297	1.082
$n=25$	1.000	1.238	1.065
$n=30$	1.000	1.196	1.053

从表 3 可知,对每种情况产生的 500 个实例中,α_{avg} 比值都不大于 1.2,远小于已证明的算法 A 的最坏情况界 $3-\frac{2}{m}\left(\text{当 } m=3 \text{ 时为}\frac{7}{3}\right)$,说明算法 A 对于小规模实例能得到较好的结果。

3.2 大规模实例分析

接下来进行较大规模实例的数值实验分析,算法的近似比通过最优解的下界来计算。首先分析工件加工时间对算法的影响,令机器数 $m=4$,资源数 $\lambda=4$,工件个数为 100。工件的加工时间分别从区间[0,50]、[0,100]、[0,200]、[0,300]、[0,500]中随机取数,对这 5 种不同情况进行分析,每种情况随机产生 500 个实例,具体结果如表 4 所示。

表 4 机器数 $m=4$、工件数 $n=100$、额外资源数 $\lambda=4$ 时的实验结果

加工时间	$\alpha_{\min}$	$\alpha_{\max}$	α_{avg}
[0,50]	1.004	1.413	1.054
[0,100]	1.002	1.389	1.050
[0,200]	1.006	1.517	1.118
[0,300]	1.006	1.586	1.124
[0,500]	1.006	1.706	1.122

从表 4 可知,随着工件加工时间区间的变大,$\alpha_{\min}$、$\alpha_{\max}$ 和 α_{avg} 都略有增大,但 α_{avg} 始终稳定在 1.2 以内,且最坏情况下的近似比 $\alpha_{\max}$ 也不大于 2,说明算法在不同的工件加工时间情况下体现了较好的性能和稳定性。

其次,考虑额外资源种类数对算法的影响。令机器数 $m=4$,工件的加工时间从区间[0,100]中随机产生,生成 100 个工件。分别对资源数 $\lambda=4,5,6,7,8$ 这 5 种情况进行分析,每种情况随机产生 500 个实例,具体结果如表 5 所示。

表 5 机器数 $m=4$、工件数 $n=100$、工件加工时间区间为[0,100]时的实验结果

资源数	$\alpha_{\min}$	$\alpha_{\max}$	α_{avg}
$\lambda=4$	1.002	1.389	1.050
$\lambda=5$	1.002	1.354	1.052
$\lambda=6$	1.002	1.220	1.036
$\lambda=7$	1.003	1.142	1.031
$\lambda=8$	1.002	1.138	1.029

从表 5 可知,随着资源种类数的增加,α_{avg} 呈变小的趋势且越来越接近 1,最坏情况 $\alpha_{\max}$ 依然没有超过 1.5 且越来越小,说明算法在不同资源种类数增加这种更为复杂的情况下,不但不会变差,性能反而变得越好。

继续考虑工件数量对于算法的影响。令机器数 $m=4$、资源数 $\lambda=5$、工件加工时间从区间[0,100]中随机产生。分别对工件个数 $n=20$, 50, 100, 500, 1000 这 5 种情况进行分析,每种情况随机产生 500 个实例。具体结果如表 6 所示。

表 6 机器数 $m=4$、额外资源数 $\lambda=5$、工件加工时间区间为[0,100]时的实验结果

工件个数	$\alpha_{\min}$	$\alpha_{\max}$	α_{avg}
$n=20$	1.014	1.487	1.127
$n=50$	1.002	1.543	1.103
$n=100$	1.002	1.316	1.053
$n=500$	1.001	1.054	1.010
$n=1000$	1.000	1.024	1.005

从表 6 可知,随着工件数量的增加,α_{avg} 越来越接近于 1,即算法目标函数值越接近最优目标函数值,同时 $\alpha_{\min}$ 和 $\alpha_{\max}$ 也呈现变小趋势。对于工件数量这个最能体现问题规模大小的指标,本文算法体现了非常好的性能。

最后考虑不同的机器数对于算法的影响。令资源数 $\lambda=5$,工件加工时间从区间[0,100]中随机产生,工件数为 100。分别对机器数 $m=3,5,7,10,15$ 这 5 种不同情况进行分析,每种情况随机产生 500 个实例。具体结果如表 7 所示。

表 7　工件数 $n=100$、额外资源数 $\lambda=5$、工件加工时间区间为[0,100]时的实验结果

机器数	$\alpha_{\min}$	$\alpha_{\max}$	α_{avg}
$m=3$	1.001	1.062	1.015
$m=5$	1.005	1.143	1.039
$m=7$	1.002	1.255	1.078
$m=10$	1.000	1.080	1.027
$m=15$	1.000	1.082	1.026

从表 7 可知,随着机器数的增加,算法的性能并没有受到影响。同时,从表 7 中可以看到,当机器数 $m=10$ 和 $m=15$ 时,两种情况体现的算法性能相近,这是因为当机器数大于两倍的资源种类数时,机器就会被闲置。

综上可知,本文设计的算法 A 对于不同规模情况下的实例,在加工时间、工件数、额外资源数和机器数等因素扰动下,都能表现出远好于问题最坏界的平均性能,且效果稳定。在上述所有情况中,对于随机产生的 500 个实例,算法的总运行时间都小于 3 s,因此本文对算法的运行时间不作特别分析。

4　结　论

本文研究了一类工件加工需要额外资源且额外资源有两个单位可用的平行机调度问题,目标是极小化最大完工时间,其三参数表示为 $P_m\mid res\lambda 21\mid C_{\max}$,该问题尚未有近似算法。本文首先说明了该问题是 NP—难的,给出了该问题的一个整数规划模型和最优解的下界性质;其次给出了该问题的一个最坏情况界不大于 $3-2/m$ 的近似算法,并通过分析算法性质给出了相应的最坏情况界证明;最后进行数值实验来分析所设计近似算法在平均情况下的性能。数值实验结果表明,算法在大规模实例和不同参数扰动情况下都体现了优异和稳定的效果。

对于该类问题,可以针对更一般情况下的问题 $P_m\mid res\lambda\sigma\delta\mid C_{\max}$ 进行后续研究。虽然对于一般情况下问题的近似算法设计在理论上有一定的难度,但是研究该类问题最优解的性质,由此设计高效的启发式算法,依然具有重要的研究意义。

参考文献:

[1] 胡觉亮, 杨佳雯, 苏晓彤, 等. 机器带周期性维护时段的加工与运输协同排序问题[J]. 浙江理工大学学报(自然科学版), 2016, 35(6): 951-956.

[2] 王霄汉, 张霖, 任磊, 等. 基于强化学习的车间调度问题研究简述[J]. 系统仿真学报, 2021, 33(12): 2782-2791.

[3] 薛玲玲. 作业车间调度的块结构邻域搜索遗传算法[J]. 计算机集成制造系统, 2021, 27(10):2848-2857.

[4] Kellerer H, Strusevich V A. Scheduling parallel dedicated machines under a single non-shared resource [J]. European Journal of Operational Research, 2003, 147(2): 345-364.

[5] Li Y Z, Wang F, Lim A. Resource constraints machine scheduling: A genetic algorithm approach [C]//The 2003 Congress on Evolutionary Computation. Canberra, ACT, Australia: IEEE, 2003: 1080-1085.

[6] Kellerer H, Strusevich V A. Scheduling problems for parallel dedicated machines under multiple resource constraints[J]. Discrete Applied Mathematics, 2003, 133(1/2/3): 45-68.

[7] Hebrard E, Huguet M J, Jozefowiez N, et al. Approximation of the parallel machine scheduling problem with additional unit resources [J]. Discrete Applied Mathematics, 2016, 215: 126-135.

[8] Godet A, Lorca X, Hebrard E, et al. Using approximation within constraint programming to solve the parallel machine scheduling problem with additional unit resources[C]//Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20). New York: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(2): 1512-1519.

[9] Strusevich V A. Approximation algorithms for makespan minimization on identical parallel machines under resource constraints[J]. Journal of the Operational Research Society, 2021, 72(9): 2135-2146.

[10] Janssen T, Swennenhuis C, Bitar A, et al. Parallel machine scheduling with a single resource per job[EB/OL]. (2018-11-16)[2022-05-14]. https://arxiv.org/abs/1809.05009.

[11] Kaplan E H, Amir A. A fast feasibility test for relocation problems [J]. European Journal of Operational Research, 1988, 35(2): 201-206.

[12] Lin B M T, Huang H L. On the relocation problem with a second working crew for resource recycling[J]. International Journal of Systems Science, 2006, 37(1): 27-34.

[13] Kononov A V, Lin B M T. Minimizing the total weighted completion time in the relocation problem[J]. Journal of Scheduling, 2010, 13(2): 123-129.

[14] Cheng T C E, Lin B M T, Huang H L. Resource-constrained flowshop scheduling with separate resource recycling operations [J]. Computers & Operations Research, 2012, 39(6): 1206-1212.

[15] Lin B M T. Resource-constrained scheduling with optional recycling operations [J]. Computers & Industrial Engineering, 2015, 90: 39-45.

(责任编辑:康　锋)