arXiv:2611.32565v1 [cs.DS] 24 Nov 2026

在线平行机调度的改进下界

Junwen XIAO* Zhiyi TAN†

摘要

本文针对以最小化最大完工时间为目标的在线平行机调度，确立了新的下界。通过利用装箱算法估计最优完工时间，我们证明，任何在线算法的竞争比至少为 1.887111。此外，通过直接构造，我们证明了对于五台机器，下界为 1.74916。我们的结果改进了二十年前确立的最佳已知下界。

关键词： 调度、在线算法、竞争比、并行机。

1 引言

平行机调度是在线优化中的基准问题。考虑一组 n 相互独立的作业 $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$，它们需要在 $m \geq 2$ 台相同并行机上进行处理，记作 $M_1, \ldots, M_m$。所有作业在时间 0 时均可用，并按照给定顺序到达。当作业 J_j到达时，仅会公开 J_j及其前驱作业 $J_1, \ldots, J_{j-1}$的处理时间。关于后续作业的信息，包括其处理时间甚至它们是否存在，均属未知。一旦某个作业被分配到某台机器上，该决策便不可撤销。

调度通过将每个作业恰好分配给一台机器形成。机器的负载定义为分配给它的各作业处理时间之和，而最大完工时间则是所有机器负载中的最大值。平行机调度的基本目标是最小化最大完工时间。

为在线问题设计的算法称为在线算法。在线算法 A 的性能通常通过其竞争比进行评估，该竞争比定义为满足以下条件的最小数 θ：对于每个实例 $\mathcal{J}$ [15]，均有 $C^A(\mathcal{J}) \leq \theta C^*(\mathcal{J})$成立。在此，$C^A(\mathcal{J})$表示由 A产生的解的目标值，而 $C^*(\mathcal{J})$表示相应离线问题的最优目标值。如果不存在任何在线算法能够实现严格小于 θ的竞争比，则称在线问题具有下界 θ。如果某个在线算法的竞争比等于该下界，则称其为最优算法。研究在线问题的主要目标是设计具有尽可能小竞争比的算法，并建立尽可能大的下界，直至确定最优竞争比。

为在线问题建立下界通常涉及构造一族对抗实例，使得不存在任何单一在线算法能够在所有这些实例上都表现良好。这

*浙江大学数学科学学院，中华人民共和国杭州 310027。

†通讯作者。浙江大学数学科学学院，中华人民共和国杭州 310027。获国家自然科学基金委员会资助（12571340）。
tanzy@zju.edu.cn

要求将算法生成的解的目标值与离线最优解进行比较。对于我们的问题，这是一个关键步骤，并且归结为计算最优完工时间。由于计算精确的最优完工时间在计算上不可处理，因此只需推导出一个紧的上界即可。以往的方法通常通过系统地构造可行调度来推导这些上界。为便于分析，实例构造保持相对简单。常见策略是将作业划分为若干阶段，每个阶段仅包含有限的作业类型。处理时间经过精心选择，以限制在线算法，迫使其在每个阶段按照特定规则处理作业。

本文结合一种新颖的方法与计算搜索，为机器数量固定的在线平行机调度建立了新的下界。在保持传统的基于阶段的结构的同时，我们引入装箱策略，以获得更紧的最优完工时间估计。这一改进能够处理更多样的作业类型，从而使所得下界更接近精确最优值。我们显式构造实例，以证明 $m \geq 6$ 的下界，其中在 $m = 8192$ 处达到的最大下界为 1.887111。此外，对于$m = 5$，我们通过直接构造给出了新的下界 1.74916。所得实例足够简单，可以通过手工验证。

表1总结了我们所得的新下界，并将其与 [3, 12] 中已有的结果进行了比较。我们的方法所带来的改进也已明确列出。

m	5	6	7	8	9	10	11
[3] 中的下界	1.746250	1.773010	1.791030	1.800700	1.803040	1.809010	N/A
[12]中的下界	1.748335	1.774098	1.792668	1.803471	1.808961	1.814327	N/A
本文	1.749160	1.774240	1.793147	1.804701	1.814990	1.822499	1.829680
改进	0.000825	0.000142	0.000479	0.001230	0.006029	0.008172	N/A
m	12	13	14	15	16	17	18
下界在 [12]	1.825277	1.829976	1.836791	1.839335	1.842994	N/A	1.845481
本文	1.835064	1.838717	1.843699	1.848130	1.850929	1.852558	1.855025
改进	0.009787	0.008741	0.006908	0.008795	0.007935	N/A	0.009544
m	19	20	24	120	3600	8192	任意的
下界在 [12]	N/A	N/A	1.856699	1.875000	1.880000	N/A	1.880000
本文	1.857616	1.859272	1.863644	1.883184	1.887095	1.887111	1.887111
改进	N/A	N/A	0.006945	0.008184	0.007095	N/A	0.007111

表1：下界比较。

相关工作：早在在线问题这一概念被正式提出之前，Graham[8] 就提出了在线算法列表调度，后来证明它对于 $m = 2$和 $m = 3$ [6]是最优的。令人惊讶的是，尽管针对机器数量任意和固定的情况，算法和下界都在不断改进，迄今为止，列表调度仍然是平行机调度中唯一已知的最优算法。

对于任意台数机器，目前最优算法的竞争比为 1。9201，这一结果是二十多年前提出的 [7]。最近一次下界改进也已是二十多年前的事了。Albers [1] 证明下界至少为 1。852。Gormley 等人 [9] 通过计算机搜索确立了 1。85358 的下界。Rudin [12]

在其学位论文中证明了下界为 1。88。对于固定且较小的机器数量，Chen 等人 [3]确立了下界。Rudin 改进了这些下界，但只有 $m = 4$ 情形的下界 $\sqrt{3}$ 正式发表了 [13], ，之后 Braun 等人 [2]简化了证明。最近，Janke 报告了 $m = 4$ [10]的下界为 1。73246。关于在线平行机调度问题及其算法的全面概述，读者可参阅综述 [14, 16]。

装箱 是另一类重要的组合优化问题。在一维装箱问题中，我们给定一个物品序列，每件物品都有相应的尺寸。目标是将这些物品装入数量最少的箱子中，且每个箱子的容量相同。降序首次适应 （FFD）算法是求解装箱问题的基石[11]。FFD 首先按照尺寸非递增的顺序对物品进行排序，然后将每件物品分配到第一个仍有足够剩余容量的已打开箱子中。如果不存在这样的箱子，则打开一个新箱子。尽管关于 FFD 的研究源于 Johnson 的开创性工作 [11], ，但经过三十多年，D´osa[5]才完全刻画了其最坏情况性能。

一维装箱问题可以视为平行机调度的对偶问题。箱子对应机器，物品对应作业，物品尺寸对应处理时间。平行机调度固定机器数量以最小化最大完工时间，而装箱则固定最大完工时间以最小化所使用的机器数量。这种对偶性很早就在相关文献中得到认识，并被有效地用于从装箱算法推导调度算法 [4]。

本文其余部分安排如下。第 2 节详细介绍一般情形下下界的构造。第 3 节重点讨论五台机器的具体情形。最后，第 4 节给出一些总结性评论。

2 任意台数机器

本节将介绍用于证明任意台数机器问题下界的实例构造。我们首先介绍这些实例的一般框架，并解释为何可以利用该框架建立下界。随后，我们描述寻找此类实例的具体过程。最后，我们给出通过该方法得到的下界及其对应的实例。

2.1 一般框架

给定一个正整数 N、一个整数 $1 \le k < N$ 以及一个由实数组成的 N-元组 $\mathbf{r} = (r_1, r_2, \ldots, r_N)$，其中

$$1 = r_1 < r_2 < \cdots < r_N. \tag{1}$$

Let

$$\rho(\mathbf{r}) = \min_{1 \le i \le k} \frac{r_i}{\sum_{l=1}^{i} r_l}, \tag{2}$$

对于任意 $k+1 \le i \le N$ 和任意 $0 < \gamma \le \rho(\mathbf{r})$，定义

$$U_i(\gamma) = \frac{1}{\gamma} \sum_{l=1}^{i} r_l. \tag{3}$$

显然，对于任意元组 $\mathbf{r}$， $\rho(\mathbf{r}) \leq 1$，并且对于任意 i， $U_i(\gamma) \geq \sum_{l=1}^{i} r_l \geq r_i$。

为了证明 m 台机器的下界，我们构造如下实例。该实例中的作业被划分为 $N+1$ 个阶段。除最后一个阶段外，每个阶段均由 m 个作业组成，而最后一个阶段仅包含一个处理时间为 $U_N(\gamma)$ 的作业。在前 k 个阶段中，所有作业的处理时间均相同。阶段 i 中作业的处理时间为 r_i、 $i=1$、.....、 k。对于任意 $k+1 \leq i \leq N$，阶段 i中有 $Q+1$种作业，其中 Q为一个正整数，其取值将在实现过程中确定。阶段 i 中第j类作业的处理时间和作业数量分别为p_{ij} 和n_{ij}。我们有 $\sum_{j=1}^{Q+1} n_{ij} = m$，而$n_{ij}$ 的确切取值将稍后确定。注意，对于某些 j，可能有 $n_{ij}=0$，在这种情况下，该类型的作业实际上并不存在。这么做仅是为了记号上的方便。对于任意 $k+1 \leq i \leq N$和 $1 \leq j \leq Q+1$，令

$$p_{ij} = r_i + \frac{j-1}{Q}(U_i(\gamma) - r_i). \tag{4}$$

显然，对于任意 $k+1 \leq i \leq N$，

$$r_i = p_{i1} < p_{i2} < \cdots < p_{iQ} < p_{i,Q+1} = U_i(\gamma). \tag{5}$$

现在我们给出一个引理，说明优秀的在线算法何时必须将每个阶段中的作业分配到不同机器上。对于任意满足 $1 \leq i \leq k$ 和 $j=1$的 i，以及任意满足 $k+1 \leq i \leq N$ 和 $1 \leq j \leq Q+1$的 i，令 $\mathcal{J}_{ij}$ 为由前 $i-1$ 个阶段的作业以及阶段 i 中前 j 种类型的作业组成的实例，并且

$$B_{ij}(\gamma) = \frac{\sum_{l=1}^{i} r_l + p_{ij}}{1+\gamma}. \tag{6}$$

根据实例的定义，不难看出

$$C^*(\mathcal{J}_{i1}) = \sum_{l=1}^{i} r_l, \quad 1 \leq i \leq k. \tag{7}$$

然而，对于其余实例 $\mathcal{J}_{ij}$， $C^*(\mathcal{J}_{ij})$ 无法用处理时间表示为简单的表达式。在这种情况下， $B_{ij}(\gamma)$ 是 $C^*(\mathcal{J}_{ij})$ 的良好替代。

引理 2.1 假设

$$C^*(\mathcal{J}_{ij}) \leq B_{ij}(\gamma), \quad \textit{for all } k+1 \leq i \leq N,\ 1 \leq j \leq Q+1 \textit{ with } n_{ij} > 0. \tag{8}$$

任何竞争比小于 $1+\gamma$ 的算法 A，其中 $0 < \gamma \leq \rho(\mathbf{r})$，对于所有 $1 \leq i \leq N$，都必须将第 i 阶段的作业分配到不同机器上。

证明。 假设算法 A 的竞争比小于 $1+\gamma$。如果算法 A 将第一阶段的至少两个作业分配给同一台机器，那么 $C^A \geq 2r_1$ 和 $\frac{C^A}{C^*} \geq \frac{2r_1}{r_1} = 2 \geq 1+\gamma$。因此，第一阶段的作业必须在不同机器上处理。假设对于$2 \leq i \leq k$，前 $i-1$ 个阶段中的作业分别调度在不同机器上。如果算法将阶段 i 的至少两个作业分配给同一台机器，那么 $C^A \geq \sum_{l=1}^{i-1} r_l + 2r_i$。因此，根据（7）和（1），

$$\frac{C^A}{C^*} \geq \frac{\sum_{l=1}^{i-1} r_l + 2r_i}{\sum_{l=1}^{i} r_l} = 1 + \frac{r_i}{\sum_{l=1}^{i} r_l} \geq 1 + \rho(\mathbf{r}) \geq 1 + \gamma.$$

因此，阶段 i 的作业必须在不同机器上处理。根据归纳法，前 k 个阶段中的作业分别调度在不同机器上。

假设对于 $k+1 \le i \le N$，前 $i-1$ 个阶段中的每个阶段的作业都调度在不同机器上。如果阶段 i 的第一类作业中至少有两个被处理在同一台机器上，则 $C^A(\mathcal{J}_{i1}) \ge \sum_{l=1}^{i-1} r_l + 2p_{i1}$。根据 (8) 和 (5)，

$$\frac{C^A(\mathcal{J}_{i1})}{C^*(\mathcal{J}_{i1})} \ge \frac{\sum_{l=1}^{i-1} r_l + 2p_{i1}}{B_{i1}(\gamma)} = \frac{\sum_{l=1}^{i} r_l + p_{i1}}{B_{i1}(\gamma)} = 1+\gamma.$$

因此，阶段 i 的第一类作业必须被处理在不同机器上。假设阶段 i 中前 $j-1$ 类作业都被处理在不同机器上。如果阶段 i 中第 j 类作业被处理在与该阶段另一作业相同的机器上，则 $C^A(\mathcal{J}_{ij}) \ge \sum_{l=1}^{i-1} r_l + p_{i1} + p_{ij}$。根据 (8) 和 (5)，

$$\frac{C^A(\mathcal{J}_{ij})}{C^*(\mathcal{J}_{ij})} \ge \frac{\sum_{l=1}^{i-1} r_l + p_{i1} + p_{ij}}{B_{ij}(\gamma)} = \frac{\sum_{l=1}^{i} r_l + p_{ij}}{B_{ij}(\gamma)} = 1+\gamma.$$

因此，阶段 i 的前 j 类作业必须被处理在不同机器上。通过归纳，阶段 i 的作业被分配到不同机器上，因此前 N 个阶段中的每个阶段的作业也是如此。□

验证引理 2.1 的关键条件（8）需要计算相当多实例的最优完工时间，这在证明下界时很常见。然而，计算精确的最优完工时间在计算上是不可处理的。因此，有时会使用一个易于计算且相对紧的最优完工时间上界来替代。我们通过考虑对偶问题来处理这一问题：对于给定的最大完工时间界，最小化所需的机器数量，而不是在固定数量的机器下最小化最大完工时间。具体而言，在 m 台机器上调度一组作业的最优完工时间至多为 b，当且仅当所有作业都能在至多 m 台机器上完成调度，并且每台机器的负载至多为 b。这种重新表述使我们能够使用装箱问题文献中的算法。

下面的 FFD 程序基于装箱问题 [11] 的降序首次适应算法。给定一组作业和目标最大完工时间界 b，FFD 首先按照处理时间非递增的顺序对作业进行排序，然后依次按此顺序分配作业。每个作业都会被分配到第一台已启用的机器上，且加入该作业后其负载不超过 b。如果不存在这样的机器，则启用一台新机器并将该作业分配给它。如果 FFD 至多使用 m 台机器，则会得到一个最大完工时间不超过 b 的可行调度，这也意味着最优完工时间同样至多为 b。然而，反向结论不一定成立。如果最优完工时间至多为 b，则带有界 b 的 FFD 仍可能使用超过 m 台机器。这是基于 FFD 的方法的局限之一。

使用FFD程序，我们可以确定使(8)得到满足的n_{ij}。从阶段的第一种类型（$k+1$）开始，依次确定每个阶段中各类型作业的数量。

给定 γ 的固定值，对于每个 $k+1 \le i \le N$ 和 $1 \le j \le Q+1$，找到一个足够好的值 n_{ij}，使得 $C^*(\mathcal{J}_{ij}) \le B_{ij}(\gamma)$。该过程可按如下方式实现。首先，在最大完工时间界 $B_{ij}(\gamma)$ 下，对 $\mathcal{J}_{i,\,j-1}$ 应用 FFD程序，其中，为了记号方便，$\mathcal{J}_{k+1,\,0}$ 表示前 k 个阶段中的作业集，而 $\mathcal{J}_{i,\,0} = \mathcal{J}_{i-1,\,Q+1}$ 对于 $k+1 < i \le N$。当

过程 FFD($\mathcal{J}_0$, b)

输入： 一个作业集 $\mathcal{J}_0$，其处理时间为 a_1, a_2, ..., a_s, 以及一个上界 b。
输出： 所使用的机器数量；或者 ∞如果至少有一个作业的处理时间大于 b。

重新为作业编制索引，使得 $a_1 \geq a_2 \geq \cdots \geq a_s$。
如果 $a_1 > b$，**则**
    返回 ∞
结束条件判断
$c \leftarrow 1$; $\ell_1 \leftarrow a_1$
对于 $j = 2$、...、s **执行**
    $v \leftarrow 1$
    当 $v \leq c$ **且** $\ell_v + a_j > b$ **时执行**
        $v \leftarrow v + 1$
    结束循环
    如果 $v = c + 1$ **则**
        $c \leftarrow c + 1$; $\ell_c \leftarrow 0$
    结束如果
    $\ell_v \leftarrow \ell_v + a_j$
结束循环
返回 c

该过程使用的机器少于 m 台时，我们向 $\mathcal{J}_{i,\ j-1}$ 中加入 d个处理时间为 p_{ij}的作业，其中 d是当前未使用机器的数量，然后再次运行 FFD。当增广作业集仍然只需要至多 m 台机器时，保留这 d 个作业并重复该步骤。一旦 FFD 使用至少 m 台机器，该过程便终止。之后，继续逐个添加作业，直到所需机器数超过 m 台。在上述过程中，取所添加作业数量的最大值，且此时使用的机器数不超过 m，将其作为 n_{ij}的值。

为确保每个阶段的作业数量恰好为 m，我们会在某个阶段内作业数量达到 m 后立即停止添加作业，并将该阶段中所有后续类型的作业数量设为 0。即便如此，仍可能出现以下情况：对于某个 $k+1 \leq i \leq N$， $\sum_{j=1}^{Q+1} n_{ij} < m$，此时构造失败。如果截至第 N阶段都未发生失败，但 FFD($\mathcal{J}$, $U_N(\gamma)$) $> m$，构造同样失败。否则，构造成功。以下定理阐明了此类成功构造的实例所具有的理论意义。

定理 2.1 对于给定的 m 和 γ，如果构造成功，则不存在竞争比小于 $1+\gamma$ 的算法。

证明。 假设算法 A 的竞争比小于 $1+\gamma$。由于该构造在任何阶段 i都不会失败，因此每个阶段恰好有 m 个作业。此外，对于所有满足 $n_{ij} > 0$的 $k+1 \leq i \leq N$和$1 \leq j \leq Q+1$，根据 n_{ij}的选择，FFD($\mathcal{J}_{ij}$, $B_{ij}(\gamma)$) $\leq m$，因此 $C^*(\mathcal{J}_{ij}) \leq B_{ij}(\gamma)$。根据引理 2.1，前 N 个阶段中的作业分别在不同机器上处理。因此，在最后一个作业到达后，无论该作业在哪台机器上处理， $C^A(\mathcal{J}) \geq \sum_{i=1}^{N} r_i + U_N(\gamma)$。由于该构造不会失败，FFD($\mathcal{J}$, $U_N(\gamma)$) $\leq m$，因此 $C^*(\mathcal{J}) \leq U_N(\gamma)$。因此，

根据（3），我们有

$$\frac{C^A(\mathcal{J})}{C^*(\mathcal{J})} \geq \frac{\sum_{i=1}^{N} r_i + U_N(\gamma)}{U_N(\gamma)} = 1 + \gamma,$$

这与 A 的竞争比小于 $1+\gamma$ 的假设相矛盾。 □

2.2 构造示例

在本小节中，我们通过一个简单示例说明实例构造的完整过程。假设 $m = 6$、 $N = 3$、 $k = 1$ 和 $\mathbf{r} = (1, 2, 3)$。我们固定 $\gamma = \frac{2}{3}$ 并选择 $Q = 3$。然后，根据 (3)，我们有 $U_2(\gamma) = \frac{9}{2}$ 和 $U_3(\gamma) = 9$。通过 (4) 和 (6) 获得的 p_{ij} 和 $B_{ij}(\gamma)$ 对于 $i = 2, 3$ 以及 $1 \leq j \leq 4$ 的相应值如表 2 所示。

	阶段 1		阶段 2			阶段 3			阶段 4	
j	r_1	数量	p_{ij}	B_{ij}	n_{ij}	p_{ij}	B_{ij}	n_{ij}	$U_3(\gamma)$	数量
1	1	6	2	3	6	3	$\frac{27}{5}$	4	9	1
2			$\frac{17}{6}$	$\frac{7}{2}$	0	5	$\frac{33}{5}$	1		
3			$\frac{11}{3}$	4	0	7	$\frac{39}{5}$	1		
4			$\frac{9}{2}$	$\frac{9}{2}$	0	9	9	0		

表 2：实例概览

根据实例的结构，阶段 1 包含 6 个处理时间为 $r_1 = 1$ 的作业，而阶段 4 由一个处理时间为 $U_3(\gamma) = 9$ 的作业组成。接下来，我们确定阶段 2 和阶段 3 中各类型作业的数量。

注意，FFD($\mathcal{J}_{2,\ 0}$，3) $= 2$，且阶段 2 中第一类作业的处理时间为 2。再增加 4 个作业会使 FFD 使用的机器数量增加到 5，而再增加 1 个作业则会使该数量增加到 6。即使再加入这样一个作业，FFD 仍然需要 6 台机器。由于此时阶段 2 已经包含 6 个作业，因此我们有 $n_{21} = 6$and $n_{22} = n_{23} = n_{24} = 0$。因此，阶段 2 中每种类型的作业数量已完全确定。

类似地，FFD($\mathcal{J}_{3,\ 0,\ 27\ 5}$) $= 4$，且阶段 3 中第一类作业的处理时间为 3。再增加 2 个作业会使 FFD 使用的机器数量增加到 5，而再增加 1 个作业则会使该数量增加到 6。然而，再增加 1 个作业后，FFD 需要 7 台机器。因此， $n_{31} = 4$。注意，FFD($\mathcal{J}_{3,\ 1,\ 33\ 5}$) $= 5$，且阶段 3 中第二类作业的处理时间为 5。再增加 1 个作业会使机器数量增加到 6。然而，再增加 1 个作业后，FFD 需要 7 台机器。因此， $n_{32} = 1$。类似地，FFD($\mathcal{J}_{3,\ 2,\ 39\ 5}$) $= 5$，且阶段 3 中第三类作业的处理时间为 7。再增加 1 个作业会使机器数量增加到 6。然而，再增加 1 个作业后，FFD 需要 7 台机器。因此，$n_{33} = 1$。由于阶段 3 已经包含 6 个作业，因此 $n_{34} = 0$。所以，阶段 3 中每种类型的作业数量已完全确定。

最后，在将最后一个作业添加到作业集后，FFD($\mathcal{J}$，9) $= 6$。因此，该构造成功。

表3报告　　FFD 在此过程中生成的作业集调度。第一列

显示用于确定每个阶段各类型作业数量的初始作业集，第二列表示在之前的作业集基础上逐步添加的此类作业数量。最后一列报告 FFD 使用的机器数量。如果该数量达到 7，则不会实际将该作业添加到作业集中。

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	
$\mathcal{J}_{2,0}$	0	$\{1,1,1\}$	$\{1,1,1\}$						2
	4	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{1,1\}$			5
	1	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{1\}$		6
	1	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$	$\{2,1\}$		6
$\mathcal{J}_{3,0}$	0	$\{2,2,1\}$	$\{2,2,1\}$	$\{2,2,1\}$	$\{1,1,1\}$				4
	2	$\{3,2\}$	$\{3,2\}$	$\{2,2,1\}$	$\{2,2,1\}$	$\{1,1,1,1\}$			5
	1	$\{3,2\}$	$\{3,2\}$	$\{3,2\}$	$\{2,2,1\}$	$\{2,1,1,1\}$	$\{1,1\}$		6
	1	$\{3,2\}$	$\{3,2\}$	$\{3,2\}$	$\{3,2\}$	$\{2,2,1\}$	$\{1,1,1,1,1\}$		6
	1	$\{3,2\}$	$\{3,2\}$	$\{3,2\}$	$\{3,2\}$	$\{3,2\}$	$\{2,1,1,1,1\}$	$\{1,1\}$	7
$\mathcal{J}_{3,1}$	0	$\{3,3\}$	$\{3,3\}$	$\{2,2,2\}$	$\{2,2,2\}$	$\{1,1,1,1,1,1\}$			5
	1	$\{5,1\}$	$\{3,3\}$	$\{3,3\}$	$\{2,2,2\}$	$\{2,2,2\}$	$\{1,1,1,1,1\}$		6
	1	$\{5,1\}$	$\{5,1\}$	$\{3,3\}$	$\{3,3\}$	$\{2,2,2\}$	$\{2,2,2\}$	$\{1,1,1,1\}$	7
$\mathcal{J}_{3,2}$	0	$\{5,2\}$	$\{3,3,1\}$	$\{3,3,1\}$	$\{2,2,2,1\}$	$\{2,2,1,1,1\}$			5
	1	$\{7\}$	$\{5,2\}$	$\{3,3,1\}$	$\{3,3,1\}$	$\{2,2,2,1\}$	$\{2,2,1,1,1\}$		6
$\mathcal{J}$		$\{9\}$	$\{7,2\}$	$\{5,3,1\}$	$\{3,3,3\}$	$\{2,2,2,2,1\}$	$\{2,1,1,1,1\}$		6

表3：由FFD生成的调度方案。

2.3 过程 `Construct`

根据定理 2.1，只需成功构造这样一个实例，即可确立该下界。详细步骤在Construct过程中给出。给定机器数量 m、所选的 N 和 k的值、元组**r**以及候选下界 γ，若能成功构造实例，该过程返回 TRUE，否则返回 FALSE 。需要指出的是，由于FFD仅给出实例最优值的估计，而非精确计算，返回值 FALSE并不必然意味着下界小于 γ。因此，Construct($m, N, k, \mathbf{r}\ \gamma$)返回 FALSE 这一事实，并不能保证对任意 $\gamma' > \gamma$，Construct($m, N, k, \mathbf{r}\ \gamma'$)也会返回 FALSE 。

在整个过程中，我们用 $\{a\}^i$ 表示包含 i 个 a副本的多重集，用 $P_1 \uplus P_2$表示两个多重集 P_1 与 P_2的多重集并。对每个阶段 i， P_{ij} 表示 $\mathcal{J}_{ij}$中作业的处理时间多重集， P_{i0} 表示前 $i-1$ 个阶段中作业的处理时间多重集。此外， C_i 统计阶段 i 中已被纳入该实例的作业数量。

Construct过程不仅确定能否构造所需的实例，而且在成功时还会生成多重集 P，其中包含该实例中所有作业的处理时间。

Construct过程(m, N, k, $\mathbf{r}$ γ)**输入：** 机器数量 m、整数 N 和 k，一个 N -元组 $\mathbf{r} = (r_1, \ldots, r_N)$，其中 $1 = r_1 < r_2 < \cdots < r_N$，以及一个上界 γ。**输出：** TRUE 如果构造成功，则为 FALSE 否则 se.

1: 如果 $\gamma > \rho$**(r) 则 2: 返回** FALSE**3: 结束如果 4:** $P_{k+1,0} \leftarrow \biguplus_{l=1}^{k} \{r_l\}^{m}$。 **5: 对于** $i = k+1, \ldots, N$**执行 6:** $U_i(\gamma) \leftarrow \frac{1}{\gamma} \sum_{l=1}^{i} r_l$**7:** $C_i \leftarrow 0$**；为所有** $j = 1, \ldots, Q+1$**设置** $n_{ij} \leftarrow 0$。 **8: 对于** $j = 1, \ldots, Q+1$ **执行 9:** $P_{ij} \leftarrow P_{i,j-1}$。 **10: 如果** $C_i = m$ **则 11: 为** $\ell = j+1, \ldots, Q+1$**设置** $P_{i\ell} \leftarrow P_{ij}$**；跳出 12: 结束如果**

13 $p_{ij} \leftarrow r_i + \frac{j-1}{Q} U_i(\gamma) - r_i$ $B_{ij}(\gamma) \leftarrow \frac{\sum_{l=1}^{i} r_l + p_{ij}}{1+\gamma}$**: (( )),**

14 $= \infty$

( ) : 如果 FFD($P_{ij}, B_{ij}(\gamma)$**) 则 15: 返回** FALSE**16: 结束如果 17: 当** $C_i < m$ **时执行 18:** $d \leftarrow$ **最小**$\{m - C_i,$ **最大**$\{0, m -$ **FFD(**$P_{ij}, B_{ij}(\gamma)$**)**$\}\}$。 **19: 如果** $d > 0$ **且 FFD(**$P_{ij} \uplus \{p_{ij}\}^{d}, B_{ij}(\gamma)) \leq m$ **则 20:** $P_{ij} \leftarrow P_{ij} \uplus \{p_{ij}\}^{d}$**；** $n_{ij} \leftarrow n_{ij} + d$**；** $C_i \leftarrow C_i + d$。 **21: 否则 22: 跳出 23: 结束如果 24: 结束当 25: 当** $C_i < m$ **且 FFD(**$P_{ij} \uplus \{p_{ij}\}, B_{ij}(\gamma)) \leq m$ **时执行 26:** $P_{ij} \leftarrow P_{ij} \uplus \{p_{ij}\}$**；** $n_{ij} \leftarrow n_{ij} + 1$**；** $C_i \leftarrow C_i + 1$。 **27: 结束当 28: 结束对于 29: 如果** $C_i < m$ **则 30: 返回** FALSE**31: 结束如果 32:** $P_{i+1,0} \leftarrow P_{i,Q+1}$ **33: 结束对于 34: 如果 FFD(**$P_{N,Q+1} \uplus \{U_N(\gamma)\}, U_N(\gamma)) > m$ **则 35: 返回** FALSE**36: 结束如果 37:** $P \leftarrow P_{N,Q+1} \uplus \{U_N(\gamma)\}$。 **38: 返回** TRUE

2.4 二分搜索

构造的成功不仅取决于 N、 k 和 $\mathbf{r}$ 的选择，还取决于 γ 的选择。迄今为止，尚无判据能够识别出必然导致成功的这些参数组合。因此，我们采用二分搜索来确定 γ，以找到一个使构造成功的满意 γ^* 值。

我们通过设置 $\gamma = \rho(\mathbf{r})$ 来初始化搜索。如果构造成功，则 $\rho(\mathbf{r})$ 就是所需的值。否则，我们将 $\gamma_{LB} = \gamma_0$ 和 $\gamma_{UB} = \rho(\mathbf{r})$ 分别设为搜索区间的下界和上界，其中 γ_0 可以设为已知的下界，或者简单地设为某个我们希望所得下界能够超过的值。在每次迭代中，我们设置 $\gamma = \frac{1}{2}(\gamma_{LB} + \gamma_{UB})$，并测试构造是否成功。如果成功，则设置 $\gamma_{LB} = \gamma$。否则，设置 $\gamma_{UB} = \gamma$。当差值 $\gamma_{UB} - \gamma_{LB}$ 低于规定的阈值 δ 时，过程终止。

完整过程见 BinarySearch过程。给定机器数量 m、 N 和 k 的选定值、元组 $\mathbf{r}$ 以及一个已知下界 γ_0，该过程返回我们的方法能够证明的最大值 $\gamma^* > \gamma_0$，将其作为改进后的下界；如果找不到这样的值，则返回$\perp$。然而，由于 Construct 返回 FALSE 并不意味着相应的下界不存在，因此，对于给定参数 N、 k 和 $\mathbf{r}$，返回的 γ^* 不一定是可能达到的最佳下界。此外，返回 $\perp$也不能排除下界可以进一步改进的可能性。

过程 BinarySearch(m, N, k, $\mathbf{r}$ γ_0)

输入： 机器数量 m、整数 N 和 k、一个 N 元组 $\mathbf{r} = (r_1, \ldots\ldots, r_N)$，其中 $1 = r_1 < r_2 < \cdots < r_N$，以及一个初始下界 γ_0。
输出： 由构造证明的下界 $\gamma^* > \gamma_0$ 或 $\rho(\mathbf{r})$，或者 $\perp$ 如果没有更大的值比 γ_0 成功。

如果 Construct(m, N, k,$\mathbf{r}$ γ_0) $=$ FALSE **则**
    返回 $\perp$
结束如果
如果 Construct(m, N, k,$\mathbf{r}$ $\rho(\mathbf{r})$) $=$ TRUE **则**
    返回 $\rho(\mathbf{r})$
结束如果
$\gamma_{LB} \leftarrow \gamma_0$; $\gamma_{UB} \leftarrow \rho(\mathbf{r})$; $\gamma^* \leftarrow \perp$
当 $\gamma_{UB} - \gamma_{LB} > \delta$ **时**
    $\gamma \leftarrow \frac{1}{2}(\gamma_{LB} + \gamma_{UB})$
    如果 Construct(m, N, k, $\mathbf{r}$ γ) $=$ TRUE **那么**
        $\gamma_{LB} \leftarrow \gamma$; $\gamma^* \leftarrow \gamma$
    else
        $\gamma_{UB} \leftarrow \gamma$
    结束 if
结束 while
返回 γ^*

2.5 初始下界和局部搜索

在前一小节中，我们介绍了一般框架，留下了许多未指定的参数。在本小节中，我们明确这些参数的取值，重点说明元组 **r** 的构造。我们首先给出固定 N 和 k 时，初始选择的 **r** 的精确取值。如果该选择无法得到令人满意的下界，则采用局部搜索来获得更优的元组。

令 $\mu \in (0, 1)$ 为一个参数，其值将在稍后确定。令

$$r_i = \frac{\mu}{(1-\mu)^{i-1}} r_1, \quad 2 \le i \le k, \tag{9}$$

and

$$r_i = \frac{2\mu r_{i-1} - \sum_{l=1}^{i-1} r_l}{1-\mu}, \quad k+1 \le i \le N. \tag{10}$$

对于 $2 \le i \le k$，根据 (9),

$$\frac{r_i}{\sum_{l=1}^{i} r_l} = \frac{\frac{\mu}{(1-\mu)^{i-1}} r_1}{\left(1 + \sum_{l=2}^{i} \frac{\mu}{(1-\mu)^{l-1}}\right) r_1} = \frac{\frac{\mu}{(1-\mu)^{i-1}}}{1 + \frac{\frac{\mu}{1-\mu}\left(1 - \frac{1}{(1-\mu)^{i-1}}\right)}{1 - \frac{1}{1-\mu}}} = \mu.$$

因此，对于 $1 < k < N$， $\rho(\mathbf{r}) = \mu$；对于 $k = 1$， $\rho(\mathbf{r}) = 1$。令 $\gamma = \mu$，最后一个作业的处理时间变为

$$U_N(\mu) = \frac{1}{\mu} \sum_{l=1}^{N} r_l = \frac{1}{\mu} \left(2\mu r_{N-1} - (1-\mu) r_N + r_N\right) = r_N + 2r_{N-1}$$

根据 (10)。因此， $C^*(\mathcal{J}) \ge r_N + 2r_{N-1}$。另一方面，阶段 N 中有 m 个作业，其处理时间至少为 r_N，而最后一个作业的处理时间为 $U_N(\gamma) > r_N$。因此，$C^*(\mathcal{J}) \ge 2r_N$。所以，按照竞争分析中的标准技术，我们选择 μ 使得 $r_N = 2r_{N-1}$，这有助于使最优调度中的机器负载大致相等。在此条件下，只要 $\gamma > \mu$， $U_N(\gamma) < U_N(\mu) = 2r_N \le C^*(\mathcal{J})$。由于构造成功的唯一条件是 FFD($\mathcal{J}$, $U_N(\gamma)$) $\le m$，因此只有当 $\gamma \le \mu$ 时才可能实现，即使在 $k = 1$ 的情况下也是如此。

我们选择 N 和 k，使得 $1 + \mu$ 的相应值成为下界的候选值。**r** 的元组可由（9）和（10）得到。给定一个值 γ_0，BinarySearch过程($m, N, k,$ **r** γ_0)尝试寻找严格优于 γ_0的下界。如果该过程未能找到这样的下界，则可以更改 N 和 k 的值，并使用新的参数重复上述过程。如果该过程找到了这样的下界，但结果并不理想，则可以更改 N 和 k 的值并重复上述过程，或者保持它们不变，应用下文所述的局部搜索方法，通过扰动**r**来尝试获得更好的下界。

对于给定的 N和 k，令 $\mathbf{r}^{(0)} = (r_1^{(0)}, \ldots\ldots, r_N^{(0)})$ 和 $\gamma^{(0)}$ 分别表示由 (9) 和 (10) 得到的初始元组及相应的下界，作为当前运行的起点。不过，初始元组和下界并不局限于这些值。或者，也可以将该过程前一次运行得到的元组和下界作为下一次运行的起点。

假设在第 t 次迭代中得到的参数为 $\mathbf{r}^{(t)}$ 和 $\gamma^{(t)}$。选择步长 h，取 $\boldsymbol{\varepsilon}=(0,\varepsilon_2,\ldots\ldots,\varepsilon_N)$，其中对于 $i=2,\ldots\ldots,N$，有 $|\varepsilon_i|\leq h$，并令 $\mathbf{r}=(r_1,r_2,\ldots\ldots,r_N)$，其中对于 $i=2,\ldots\ldots,N$，有$r_i=r_i^{(t-1)}(1+\varepsilon_i)$。如果 $\mathbf{r}$ 不满足 (1)，则该元组不可行。在这种情况下，$\mathbf{r}$ 被拒绝，因此需要重新生成 $\boldsymbol{\varepsilon}$，并进入下一次迭代。如果 $\mathbf{r}$ 可行，则应用 BinarySearch$(m,N,k,\mathbf{r}\ \gamma^{(t-1)})$。如果它返回 $\perp$，表示找不到优于 $\gamma^{(t-1)}$ 的下界，甚至根本无法获得有效的下界，则拒绝元组 $\mathbf{r}$。这就需要重新生成 $\boldsymbol{\varepsilon}$，并进入下一次迭代。否则，令 $\mathbf{r}^{(t)}=\mathbf{r}$ 和 $\gamma^{(t)}=\gamma$，然后进入下一次迭代。迭代在经过 T 步后终止。如果截至此时仍未找到令人满意的下界，则必须重新选择 N和 k。

下面给出局部搜索过程。过程中涉及的步长 h 和迭代次数 T可以根据实际需要进行选择。每个 ϵ_i 均从 $[-h,h]$上的均匀分布中抽取，从而可以通过随机扰动获得不同的元组。给定局部搜索过程 LocalSearch$(m,N,k,\mathbf{r}^{(0)},\gamma^{(0)})$的输出 γ^{**} 和$\mathbf{r}^{**}$，运行 Construct$(m,N,k,\mathbf{r}^{**},\gamma^{**})$过程即可得到相应的实例。

过程 局部搜索$(m,N,k,\mathbf{r}^{(0)},\gamma^{(0)})$

输入：机器数量 m、整数 N 和 k、初始下界 $\gamma^{(0)}$ 以及相应的 N-元组 $\mathbf{r}^{(0)}=(r_1^{(0)},\ldots,r_N^{(0)})$，其中 $1=r_1^{(0)}<r_2^{(0)}<\cdots<r_N^{(0)}$。输出：下界 γ^{**} 及相应的元组 $\mathbf{r}^{**}$。1：对于 $t=1,\ldots,T$执行 2：为所有 $i\in\{2,\ldots,N\}$绘制 $\epsilon_i\sim$Uni$[-h,h]$ 3：设置 $r_1\leftarrow 1$ 和 $r_i\leftarrow r_i^{(t-1)}(1+\epsilon_i)$，其中 $2\leq i\leq N$ 4：如果不满足 $r_1<r_2<\cdots<r_N$，则 5：$\mathbf{r}^{(t)}\leftarrow\mathbf{r}^{(t-1)}$；$\gamma^{(t)}\leftarrow\gamma^{(t-1)}$ 6：继续 7：结束 if 8：$\mathbf{r}\leftarrow(r_1,r_2,\ldots,r_N)$ 9：$\gamma\leftarrow$ BinarySearch$(m,N,k,\mathbf{r}\ \gamma^{(t-1)})$ 10：如果 $\gamma\neq\perp$，则 11：$\mathbf{r}^{(t)}\leftarrow\mathbf{r}$；$\gamma^{(t)}\leftarrow\gamma$ 12：否则 13：$\mathbf{r}^{(t)}\leftarrow\mathbf{r}^{(t-1)}$；$\gamma^{(t)}\leftarrow\gamma^{(t-1)}$ 14：结束 if 15：结束 for 16：$\mathbf{r}^{**}\leftarrow\mathbf{r}^{(T)}$；$\gamma^{**}\leftarrow\gamma^{(T)}$ 17：返回 γ^{**} 和 $\mathbf{r}^{**}$

2.6 计算结果

在本小节中，我们总结通过实现上述方法得到的下界。我们首先说明程序中所用参数的选取原则。这些选择大多基于经验。较大的 Q、h 和 T 会增加找到能产生更大下界的实例的可能性，

但这并不能保证一定如此，同时也会使计算耗时更长。剩余参数 δ 固定为 10^{-8} ，以确保达到所需的准确度。

对于每个固定的 m，我们从较大的 h 和较小的 Q 与 T 值开始，然后逐步增加 Q 和 T，或减小 h，所使用的参数值取自 $Q \in \{100, 1000, 2048, 10000\}$,

$$h \in \{0.3, 0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001, 0.000003\},$$

以及 $T \in \{10, 20, 30, 100, 200, 500, 1000, 1500, 2000, 5000\}$。

所有计算均在一台配备 AMD Ryzen 7 8745H CPU（16 个逻辑处理器，3.8 GHz 基准频率）和 24 GB 内存、运行 Windows 11 家庭版的笔记本电脑上完成。算法使用 Python 3.13.2 实现。每次运行耗时在 10 秒至 3 小时之间。例如，对于 $m = 6$、 $Q = 1000$、 $h \in \{0.3, 0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001\}$ 和 $T = 1500$，单次运行耗时约 10 至 50 秒，八次运行总计约 200 秒。对于 $m = 2048$、 $Q \in \{100, 1000, 2048\}$、 $h \in \{0.01, 0.001, 0.0001, 0.00003, 0.00001\}$ 和 $T = 10$，15 种参数组合预计总共需要 2 至 3 小时，平均每次运行约 10 分钟。

表4总结了我们的方法所得到的下界，相应的实例见补充材料 [17]。从表中可以看出，当 m 相对较大时，初始下界已经相当不错；而局部搜索带来了显著改进，尤其是当 m相对较小时。

表5展示了局部搜索开始时的前几个步骤，起点是 $m = 6$、 $N = 8$和 $k = 1$情况下的初始下界。参数 h 设置为 0.3。列出了每一步获得的元组，并将每一步归类为“Accept”或“Reject”。在“Accept”步骤中，可以得到更好的下界。

3 五台机器的下界

本节给出一个直接构造的实例，说明五台相同机器上的在线调度下界至少为 1.74916。

定义

$$f(x) = 16x^4 - 46x^3 - 32x^2 - 13x + 42. \tag{11}$$

对于 $0 < x < 1$,

$$f'(x) = 64x^3 - 138x^2 - 64x - 13 < -74x^2 - 64x - 13 < 0.$$

因此， $f(x)$ 在 (0，1) 上单调递减。注意,

$$f\left(\frac{7491}{10000}\right) = \frac{4154052220311}{625000000000000} > 0$$

and

$$f\left(\frac{7492}{10000}\right) = -\frac{21983208093}{4882812500000} < 0,$$

由此可知， $f(x)$ 在 (0，1) 中有唯一根 β，其满足

$$\frac{7491}{10000} < \beta < \frac{7492}{10000}. \tag{12}$$

m	N	k	μ	$1+\gamma^*$	$1+\gamma^{**}$	m	N	k	μ	$1+\gamma^*$	$1+\gamma^{**}$
6	8	1	0.876990	1.744366	1.774240	27	13	3	0.883055	1.858084	1.866473
7	7	1	0.873729	1.768565	1.793147	28	12	3	0.882016	1.858263	1.867420
8	8	2	0.876406	1.784615	1.804701	29	10	2	0.880669	1.861721	1.868283
9	7	1	0.873729	1.795486	1.814990	40	8	1	0.876990	1.870433	1.872611
10	7	1	0.873729	1.802418	1.822499	50	8	1	0.876990	1.873609	1.874874
11	10	2	0.880669	1.807120	1.829680	54	9	2	0.878874	1.874902	1.876409
12	12	1	0.883237	1.822577	1.835064	68	10	1	0.880976	1.877163	1.879442
13	7	1	0.873729	1.829957	1.838717	72	10	2	0.880669	1.878142	1.879446
14	7	2	0.872871	1.829689	1.843699	92	11	2	0.882016	1.880028	1.881068
15	7	1	0.873729	1.832254	1.848130	99	12	2	0.883055	1.880354	1.881378
16	9	2	0.878874	1.837561	1.850929	108	12	1	0.883237	1.881112	1.882274
17	10	2	0.880669	1.843359	1.852558	112	13	2	0.883873	1.881179	1.882715
18	14	1	0.884647	1.844327	1.855025	120	14	2	0.884530	1.881492	1.883184
19	11	3	0.880669	1.849440	1.857616	128	14	2	0.884530	1.881786	1.883205
20	12	3	0.882016	1.850440	1.859272	256	16	1	0.885586	1.884353	1.884730
21	13	3	0.883055	1.851633	1.860774	512	17	1	0.885942	1.885906	1.885915
22	13	3	0.883055	1.853224	1.862341	1024	20	2	0.886779	1.886274	1.886441
23	12	3	0.882016	1.855232	1.863082	2048	24	1	0.887357	1.884898	1.886851
24	11	2	0.882016	1.856041	1.863644	3600	21	1	0.886913	1.878676	1.887095
25	12	3	0.882016	1.856699	1.864555	4096	21	1	0.886913	1.878676	1.887095
26	12	3	0.882016	1.857110	1.865408	8192	21	1	0.886913	1.878676	1.887111

表4：不同数量机器的下界。

Step	r_2	r_3	r_4	r_5	r_6	r_7	r_8	γ	决策	γ^*
0	6.129	29.440	122.493	453.522	1486.736	4132.853	8265.706	–	初始	0.744366
1	4.414	28.401	139.109	319.146	1373.938	3681.216	10268.889	0.770099	接受	0.770099
2	4.241	34.514	113.191	238.785	1249.996	2665.059	7972.625	–	拒绝	0.770099
3	3.119	23.855	145.890	317.966	1744.028	3796.903	11167.705	–	拒绝	0.770099
4	5.359	27.397	129.783	378.602	1197.137	3599.893	9883.748	–	拒绝	0.770099
5	3.991	32.940	126.130	268.878	1092.924	4063.684	10272.140	0.772073	接受	0.772073
6	3.375	33.495	108.270	324.202	1004.438	5000.522	11899.926	–	拒绝	0.772073
7	4.983	28.440	126.244	293.293	1057.580	4456.927	10968.492	–	拒绝	0.772073
8	2.891	29.571	97.559	304.897	922.656	4069.974	9976.267	–	拒绝	0.772073
9	3.204	31.632	109.279	251.674	1319.090	5221.449	12749.013	–	拒绝	0.772073
10	3.765	36.024	103.060	341.795	1327.113	3049.875	8936.257	–	拒绝	0.772073
11	3.940	38.542	135.746	289.555	941.115	3221.358	8511.047	–	拒绝	0.772073
12	4.410	39.115	156.014	339.641	1145.220	5230.967	12710.487	–	拒绝	0.772073
13	3.261	35.498	90.042	245.210	1239.478	4803.654	11794.881	–	拒绝	0.772073
14	4.812	40.464	156.266	235.115	1190.418	4464.485	11196.546	–	拒绝	0.772073
15	3.252	33.014	105.268	295.230	1191.910	4344.718	10958.238	0.772598	接受	0.772598
16	3.873	26.300	101.130	323.598	1013.381	5020.555	11903.667	–	拒绝	0.772598
17	2.362	34.093	80.883	312.901	1111.551	3994.904	10157.245	–	拒绝	0.772598
18	3.224	24.264	85.429	264.461	983.159	3048.948	8089.718	–	拒绝	0.772598
19	3.881	34.090	115.527	213.399	951.002	3994.217	9745.323	–	拒绝	0.772598
20	2.672	33.956	94.321	264.157	1249.828	4953.275	12104.281	–	拒绝	0.772598

表5： $m = 6$的局部搜索的前几个步骤。

更准确地说， $\beta \approx 0.7491596$。

下面，我们定义一些与 β 相关的量。随后，我们将陈述一个引理，给出这些量之间的若干等式和不等式，其证明推迟至附录。

Let

$$s_1 = 15\beta(2\beta-1) \approx 5.59981, \tag{13}$$

$$s_0 = \frac{6-4\beta}{5\beta}s_1 = 3(2\beta-1)(6-4\beta) = -24\beta^2+48\beta-18 \approx 4.48990, \tag{14}$$

$$s_2 = \frac{1}{2\beta-1}(s_0+s_1) = \frac{6+\beta}{5\beta(2\beta-1)}s_1 = 3(6+\beta) = 3\beta+18 \approx 20.24748, \tag{15}$$

$$\begin{aligned} t_1 &= (2\beta+1)s_1 - s_0 = 15\beta(2\beta-1)(2\beta+1) - (-24\beta^2+48\beta-18) \\ &= 60\beta^3+24\beta^2-63\beta+18 \approx 9.50021, \end{aligned} \tag{16}$$

$$\begin{aligned} t_2 &= 4(s_2-s_1) - 5s_0 - t_1 \\ &= 4\left(3\beta+18-15\beta(2\beta-1)\right) - 5(-24\beta^2+48\beta-18) - (60\beta^3+24\beta^2-63\beta+18) \\ &= -60\beta^3-24\beta^2-105\beta+144 \approx 26.64097, \end{aligned} \tag{17}$$

$$r_4 = \beta s_0 = \frac{6-4\beta}{5}s_1 = 3\beta(2\beta-1)(6-4\beta) = -24\beta^3+48\beta^2-18\beta \approx 3.36365, \tag{18}$$

$$\begin{aligned} r_3 &= \frac{1}{3}(2(s_2-s_1) - 5r_4 - t_1) \\ &= \frac{1}{3}\left(2\left(3\beta+18-15\beta(2\beta-1)\right) - 5(-24\beta^3+48\beta^2-18\beta) - (60\beta^3+24\beta^2-63\beta+18)\right) \\ &= 20\beta^3-108\beta^2+63\beta+6 \approx 0.99229, \end{aligned} \tag{19}$$

$$\begin{aligned} r_2 &= \frac{1}{5}s_1 - r_3 = 3\beta(2\beta-1) - (20\beta^3-108\beta^2+63\beta+6) \\ &= -20\beta^3+114\beta^2-66\beta-6 \approx 0.12767, \end{aligned} \tag{20}$$

$$\begin{aligned} r_1 &= s_0 - r_2 - r_3 - r_4 = s_0 - \frac{1}{5}s_1 - r_4 \\ &= (-24\beta^2+48\beta-18) - 3\beta(2\beta-1) - (-24\beta^3+48\beta^2-18\beta) \\ &= 24\beta^3-78\beta^2+69\beta-18 \approx 0.00629. \end{aligned} \tag{21}$$

引理 3.1 (*i*) $\frac{r_2}{r_1+r_2} \geq \beta$。 ***(ii)*** $\frac{r_3}{r_1+r_2+r_3} \geq \beta$。 ***(iii)*** $2r_4+3r_3 < 3r_4+r_3 < t_1+r_3+5r_2+5r_1 < 2s_1$。 ***(iv)*** $s_1+s_2 < 5r_4+t_1$。 ***(v)*** $\frac{s_0+s_1+2s_2}{s_0+4r_4+t_1} \geq 1+\beta$。 ***(vi)*** $t_2+r_4+2r_3 < s_2+t_1+2r_3+4r_2+4r_1 < 2s_1+s_2+r_3+r_2+r_1 < s_1+s_2+2r_4$。 ***(vii)*** $\frac{s_0+s_1+s_2+t_2}{s_1+s_2+2r_4} = 1+\beta$。

我们现在给出用于证明5台机器下界的实例构造。虽然该实例与一般情形具有相似的结构，但所有作业的处理时间均可显式指定。

令 $N=6$ 和 $k=4$。对于每个 $1 \leq i \leq 4$，将阶段 i 中的作业记为 R_i，其处理时间为 r_i。在接下来的两个阶段中，有 4 个处理时间相同的作业，以及 1 个处理时间更长的作业。对于 $i=5$，6，令 S_{i-4} 和 T_{i-4} 分别表示阶段 {v?} 中处理时间较短和

较长的作业，其处理时间分别为 s_{i-4} 和 t_{i-4}。最后，令 F 表示最后一个作业，其处理时间为 $2s_2$。

与引理 2.1 类似，我们证明，要使算法的竞争比小于 $1+\beta$，同一阶段内的所有作业都必须分配到不同机器上。具体而言，对于每个 $1 \le i \le 4$，如果算法 A 将前 $i-1$ 个阶段内的作业分配到不同机器上，却未能将阶段 i 中的作业分配到不同机器上，那么 $C^A(\mathcal{J}_{i1})\, C^*(\mathcal{J}_{i1}) \ge 1+\beta$。对于每个 $5 \le i \le 6$，如果算法 A 将前 $i-1$ 个阶段内的作业分配到不同机器上，却未能将阶段 i 中的较小作业分配到不同机器上，那么 $C^A(\mathcal{J}_{i1})\, C^*(\mathcal{J}_{i1}) \ge 1+\beta$。此外，如果算法 A 将前 $i-1$ 个阶段内的作业以及阶段 i 中的较小作业分配到不同机器上，但阶段 i 中的较大作业与同一阶段内的另一个作业共用一台机器，那么 $C^A(\mathcal{J}_{i2})\, C^*(\mathcal{J}_{i2}) \ge 1+\beta$。因此，算法 A 的竞争比至少为 $1+\beta$。

如果阶段 1 中的作业未被分配到不同机器，那么 $C^A(\mathcal{J}_{11}) \ge 2r_1$。显然，$C^*(\mathcal{J}_{11}) = r_1$。于是 $C^A(\mathcal{J}_{11})\, C^*(\mathcal{J}_{11}) \ge \frac{2r_1}{r_1} = 2$。

如果阶段 1 中的作业被分配到不同机器，但阶段 2 中的作业没有，那么$C^A(\mathcal{J}_{21}) \ge r_1 + 2r_2$。显然，$C^*(\mathcal{J}_{21}) = r_1 + r_2$。然后根据引理 3.1(i)，

$$\frac{C^A(\mathcal{J}_{21})}{C^*(\mathcal{J}_{21})} \ge \frac{r_1 + 2r_2}{r_1 + r_2} \ge 1 + \beta.$$

如果前两个阶段中的作业分别被分配到不同机器，但阶段 3 中的作业没有，那么 $C^A(\mathcal{J}_{31}) \ge r_1 + r_2 + 2r_3$。显然，$C^*(\mathcal{J}_{31}) = r_1 + r_2 + r_3$。然后根据引理 3.1(ii)，

$$\frac{C^A(\mathcal{J}_{31})}{C^*(\mathcal{J}_{31})} \ge \frac{r_1 + r_2 + 2r_3}{r_1 + r_2 + r_3} \ge 1 + \beta.$$

如果前三个阶段中的作业分别被分配到不同机器，但阶段 4 中的作业没有，那么 $C^A(\mathcal{J}_{41}) \ge r_1 + r_2 + r_3 + 2r_4$。显然，$C^*(\mathcal{J}_{41}) = r_1 + r_2 + r_3 + r_4$。然后根据 (21) 和 (18)，

$$\frac{C^A(\mathcal{J}_{41})}{C^*(\mathcal{J}_{41})} \ge \frac{r_1 + r_2 + r_3 + 2r_4}{r_1 + r_2 + r_3 + r_4} = \frac{s_0 + r_4}{s_0} = 1 + \beta.$$

如果前四个阶段中的作业分别分配给不同机器，但第5阶段中的较小作业未如此分配，那么 $C^A(\mathcal{J}_{51}) \ge s_0 + 2s_1$。最优调度中分配给 $\mathcal{J}_{51}$ 的每台机器上的作业如下所示，

$$\{R_1, R_4, S_1\}, \{R_1, R_4, S_1\}, \{R_1, R_4, S_1\}, \{R_1, R_4, S_1\}, \{R_1, R_2, R_2, R_2, R_2, R_2, R_3, R_3, R_3, R_3, R_4\}.$$

根据 (20) 和 (21)，$r_1 + r_4 + s_1 = r_1 + 5r_2 + 5r_3 + r_4$，因此 $C^*(\mathcal{J}_{51}) = r_1 + r_4 + s_1 = s_0 + \frac{4}{5}s_1$。再根据 (14)，

$$\frac{C^A(\mathcal{J}_{51})}{C^*(\mathcal{J}_{51})} \ge \frac{s_0 + 2s_1}{s_0 + \frac{4}{5}s_1} = \frac{\frac{6-4\beta}{5\beta} + 2}{\frac{6-4\beta}{5\beta} + \frac{4}{5}} = 1 + \beta.$$

如果前四个阶段中的作业以及第5阶段中的较小作业分别分配到不同机器，但第5阶段中的较大作业与第5阶段中的另一个作业共用一台机器，那么

$C^A(\mathcal{J}_{52}) \geq s_0 + s_1 + t_1$。可行调度中分配给每台机器的作业如下所示,

$$\{S_1, S_1\}, \{S_1, S_1\}, \{R_3, R_4, R_4, R_4\}, \{R_3, R_3, R_3, R_4, R_4\}, \{R_1, R_1, R_1, R_1, R_1, R_2, R_2, R_2, R_2, R_2, R_3, T_1\}.$$

因此根据引理3.1(iii),

$$C^*(\mathcal{J}_{52}) \leq \max\{2s_1, 3r_4 + r_3, 2r_4 + 3r_3, t_1 + r_3 + 5r_2 + 5r_1\} = 2s_1.$$

然后根据(16),

$$\frac{C^A(\mathcal{J}_{52})}{C^*(\mathcal{J}_{52})} \geq \frac{s_0 + s_1 + t_1}{2s_1} = 1 + \beta.$$

如果前五个阶段中的作业分别被分配到不同机器上，但第6阶段中的那些较小作业没有被分配到不同机器上，那么 $C^A(\mathcal{J}_{61}) \geq s_0 + s_1 + 2s_2$。可行调度中分配给每台机器的作业 $\mathcal{J}_{61}$ 如下所示,

$$\{R_1, R_2, R_3, S_1, S_2\}, \{R_1, R_2, R_3, S_1, S_2\}, \{R_1, R_2, R_3, S_1, S_2\}, \{R_1, R_2, R_3, S_1, S_2\},$$

$$\{R_1, R_2, R_3, R_4, R_4, R_4, R_4, R_4, T_1\}.$$

因此根据引理3.1(iv)和(21),

$C^*(\mathcal{J}_{61})$ ≤最大$\{r_1 + r_2 + r_3 + s_1 + s_2, r_1 + r_2 + r_3 + 5r_4 + t_1\} = r_1 + r_2 + r_3 + 5r_4 + t_1 = s_0 + 4r_4 + t_1$.

根据引理3.1(v),

$$\frac{C^A(\mathcal{J}_{61})}{C^*(\mathcal{J}_{61})} \geq \frac{s_0 + s_1 + 2s_2}{s_0 + 4r_4 + t_1} \geq 1 + \beta.$$

如果前五个阶段中的作业以及第6阶段中的较小作业分别分配到不同机器上，但第6阶段中的较大作业与第6阶段中的另一个作业共用一台机器，那么$C^A(\mathcal{J}_{62}) \geq s_0 + s_1 + s_2 + t_2$。可行调度 $\mathcal{J}_{62}$ 中分配给每台机器的作业如下所示,

$$\{S_1, S_2, R_4, R_4\}, \{S_1, S_2, R_4, R_4\}, \{R_1, R_2, R_3, S_1, S_1, S_2\},$$

$$\{R_1, R_1, R_1, R_1, R_2, R_2, R_2, R_2, R_3, R_3, T_1, S_2\}, \{R_3, R_3, R_4, T_2\}.$$

因此，根据引理3.1(vi),

$C^*(\mathcal{J}_{62})$ ≤最大值$\{s_1 + s_2 + 2r_4,\ \ 2s_1 + s_2 + r_3 + r_2 + r_1,\ \ s_2 + t_1 + 2r_3 + 4r_2 + 4r_1,\ \ t_2 + r_4 + 2r_3\} = s_1 + s_2 + 2r_4$.

根据引理 3.1(vii),

$$\frac{C^A(\mathcal{J}_{62})}{C^*(\mathcal{J}_{62})} \geq \frac{s_0 + s_1 + s_2 + t_2}{s_1 + s_2 + 2r_4} = 1 + \beta.$$

因此，我们已经证明，同一阶段中的所有作业必须被分配到不同机器上。最后，当最后一个作业到达时， $C^A(\mathcal{J}) \geq s_0 + s_1 + s_2 + 2s_2$。 $\mathcal{J}$ 的最优调度中分配给每台机器的作业如下所示,

$$\{S_2, S_2\}, \{S_2, S_2\}, \{R_1, R_1, R_1, R_1, R_1, R_2, R_2, R_2, R_2, R_2, R_3, R_3, S_1, S_1, T_2\},$$

$$\{R_3, R_3, R_3, R_4, R_4, R_4, R_4, R_4, S_1, S_1, T_1\}, \{F\}.$$

根据(19)、(21)和(17)， $3r_3 + 5r_4 + 2s_1 + t_1 = 2s_2$ 以及

$$\begin{aligned} 5r_1 + 5r_2 + 2r_3 + 2s_1 + t_2 &= 5(s_0 - r_2 - r_3 - r_4) + 5r_2 + 2r_3 + 2s_1 + (4(s_2 - s_1) - 5s_0 - t_1) \\ &= -3r_3 - 5r_4 + 4s_2 - 2s_1 - t_1 = 2s_2. \end{aligned}$$

因此 $C^*(\mathcal{J}) = 2s_2$。根据（15），

$$\frac{C^A(\mathcal{J})}{C^*(\mathcal{J})} \geq \frac{s_0 + s_1 + s_2 + 2s_2}{2s_2} = 1 + \beta.$$

基于上述讨论，我们得到以下定理。

定理 3.1 对于五台并行机上的调度，不存在竞争比小于 $1 + \beta \approx 1.7491596$的在线算法。

上一节介绍的方法也可以应用于五台机器的情形。然而，它无法得到优于定理 3.1 中结果的下界。这是因为，尽管定理 3.1 中的实例仅包含有限种作业类型，但其经过精心构造，使得最优值能够得到更紧的界。随着机器数量增加，对最优值进行刻画变得极其困难，这凸显了直接构造方法的局限性。

4 结论

本文给出了关于 $m \geq 5$ 并行机上在线调度问题的新下界。对于 $m = 5$，我们直接构造对抗实例。对于较大的 m，我们利用装箱问题的算法估计最优完工时间，从而能够通过计算机搜索高效地生成相应实例。每种方法都有其自身的优势。直接构造能够对最优值给出更紧的估计，因而得到更强的下界。然而，这种方法依赖于问题特定的见解、临时技巧以及所涉及参数之间精细的数值关系。因此，很难得出一般性结论，并且随着机器数量增加，计算工作量会变得大到无法承受。

我们建立的下界不仅为设计最优算法提供了有针对性的指导，也为进一步改进下界指明了新的方向。然而，尽管已经取得了一些改进，我们仍不认为这些下界是紧的。要进一步加强这些下界，最直接的方法是对最优完工时间进行更紧确、更准确的估计。不过，最终可能还需要新的实例结构。一个主要困难在于，实例的结构可能取决于机器数量。此外，具有有限数量作业的实例能否得到与在线算法竞争比相匹配的下界，目前仍是一个开放问题。

5 附录

5.1 引理 3.1 的证明

证明。令 $h = \frac{3}{4} - \beta$、$a = \frac{8}{10000}$ 和 $b = \frac{9}{10000}$。根据 (12)，$0 < a < h < b$。

根据式 (21)、(20)、(18) 和 (14)，我们有

$$r_1 + r_2 = (24\beta^3 - 78\beta^2 + 69\beta - 18) + (-20\beta^3 + 114\beta^2 - 66\beta - 6) = 4\beta^3 + 36\beta^2 + 3\beta - 24 \quad (22)$$

and

$$r_1 + r_2 + r_3 = s_0 - r_4 = (1-\beta)s_0 = (1-\beta)(-24\beta^2 + 48\beta - 18) = 24\beta^3 - 72\beta^2 + 66\beta - 18. \quad (23)$$

(i) 根据式 (20) 和 (22)，

$$\begin{aligned} r_2 - \beta(r_1 + r_2) &= -20\beta^3 + 114\beta^2 - 66\beta - 6 - \beta(4\beta^3 + 36\beta^2 + 3\beta - 24) \\ &= -4\beta^4 - 56\beta^3 + 111\beta^2 - 42\beta - 6 \\ &= -4\left(\frac{3}{4} - h\right)^4 - 56\left(\frac{3}{4} - h\right)^3 + 111\left(\frac{3}{4} - h\right)^2 - 42\left(\frac{3}{4} - h\right) - 6 \\ &= -4h^4 + 68h^3 - \frac{57}{2}h^2 - \frac{93}{4}h + \frac{3}{64} > -4b^4 + 68a^3 - \frac{57}{2}b^2 - \frac{93}{4}b + \frac{3}{64} \\ &= \frac{64817374533439}{2500000000000000} > 0. \end{aligned}$$

因此，$\frac{r_2}{r_1 + r_2} \geq \beta$。

(ii) 根据式 (19) 和 (23)，

$$\begin{aligned} r_3 - \beta(r_1 + r_2 + r_3) &= 20\beta^3 - 108\beta^2 + 63\beta + 6 - \beta(24\beta^3 - 72\beta^2 + 66\beta - 18) \\ &= -24\beta^4 + 92\beta^3 - 174\beta^2 + 81\beta + 6 \\ &= -24\left(\frac{3}{4} - h\right)^4 + 92\left(\frac{3}{4} - h\right)^3 - 174\left(\frac{3}{4} - h\right)^2 + 81\left(\frac{3}{4} - h\right) + 6 \\ &= -24h^4 - 20h^3 - 48h^2 + \frac{261}{4}h + \frac{3}{32} > -24b^4 - 20b^3 - 48b^2 + \frac{261}{4}a + \frac{3}{32} \\ &= \frac{182388881755317}{1250000000000000} > 0. \end{aligned}$$

因此，$\frac{r_3}{r_1 + r_2 + r_3} \geq \beta$。

(iii) 根据 (18) 和 (19)，

$$\begin{aligned} r_4 - 2r_3 &= (-24\beta^3 + 48\beta^2 - 18\beta) - 2\left(20\beta^3 - 108\beta^2 + 63\beta + 6\right) = -64\beta^3 + 264\beta^2 - 144\beta - 12 \\ &= -64\left(\frac{3}{4} - h\right)^3 + 264\left(\frac{3}{4} - h\right)^2 - 144\left(\frac{3}{4} - h\right) - 12 \\ &= 64h^3 + 120h^2 - 144h + \frac{3}{2} > 64a^3 + 120a^2 - 144b + \frac{3}{2} = \frac{669178141}{488281250} > 0. \end{aligned}$$

根据 (16)、(22) 和 (18),

$$\begin{aligned}t_1+5r_2+5r_1-3r_4&=60\beta^3+24\beta^2-63\beta+18+5(4\beta^3+36\beta^2+3\beta-24)-3(-24\beta^3+48\beta^2-18\beta)\\&=152\beta^3+60\beta^2+6\beta-102\\&=152\left(\frac{3}{4}-h\right)^3+60\left(\frac{3}{4}-h\right)^2+6\left(\frac{3}{4}-h\right)-102\\&=-152h^3+402h^2-\frac{705}{2}h+\frac{3}{8}>-152b^3+402a^2-\frac{705}{2}b+\frac{3}{8}\\&=\frac{7250896149}{125000000000}>0.\end{aligned}$$

根据 (13)、(16)、(19) 和 (22),

$$\begin{aligned}2s_1-t_1-r_3-5r_2-5r_1&=30\beta(2\beta-1)-(60\beta^3+24\beta^2-63\beta+18)\\&\quad-(20\beta^3-108\beta^2+63\beta+6)-5(4\beta^3+36\beta^2+3\beta-24)\\&=-100\beta^3-36\beta^2-45\beta+96\\&=-100\left(\frac{3}{4}-h\right)^3-36\left(\frac{3}{4}-h\right)^2-45\left(\frac{3}{4}-h\right)+96\\&=100h^3-261h^2+\frac{1071}{4}h-\frac{3}{16}>100a^3-261b^2+\frac{1071}{4}a-\frac{3}{16}\\&=\frac{66221603}{2500000000}>0.\end{aligned}$$

因此, $2r_4+3r_3<3r_4+r_3<t_1+r_3+5r_2+5r_1<2s_1$。

(iv) 根据 (18)、(16)、(13) 和 (15),

$$\begin{aligned}5r_4+t_1-s_1-s_2&=5(-24\beta^3+48\beta^2-18\beta)+(60\beta^3+24\beta^2-63\beta+18)-15\beta(2\beta-1)-(3\beta+18)\\&=-60\beta^3+234\beta^2-141\beta=-60\left(\frac{3}{4}-h\right)^3+234\left(\frac{3}{4}-h\right)^2-141\left(\frac{3}{4}-h\right)\\&=60h^3+99h^2-\frac{435}{4}h+\frac{9}{16}>60a^3+99a^2-\frac{435}{4}b+\frac{9}{16}=\frac{1452151221}{3125000000}>0.\end{aligned}$$

因此, $s_1+s_2<5r_4+t_1$。

(v) 根据式 (14)、(13)、(15)、(18) 和 (16),

$$\begin{aligned}s_0+s_1+2s_2-(1+\beta)(s_0+4r_4+t_1)&=-\beta s_0+s_1+2s_2-(1+\beta)(4r_4+t_1)\\&=-\beta(-24\beta^2+48\beta-18)+15\beta(2\beta-1)+2(3\beta+18)\\&\quad-(1+\beta)(4(-24\beta^3+48\beta^2-18\beta)+(60\beta^3+24\beta^2-63\beta+18))\\&=36\beta^4-156\beta^3-99\beta^2+126\beta+18\\&=36\left(\frac{3}{4}-h\right)^4-156\left(\frac{3}{4}-h\right)^3-99\left(\frac{3}{4}-h\right)^2+126\left(\frac{3}{4}-h\right)+18\\&=36h^4+48h^3-\frac{657}{2}h^2+225h+\frac{153}{64}\\&>36a^4+48a^3-\frac{657}{2}b^2+225a+\frac{153}{64}=\frac{200809292155527}{78125000000000}>0.\end{aligned}$$

因此，$\frac{s_0+s_1+2s_2}{s_0+4r_4+t_1} \geq 1+\beta$。

(vi) 根据（15）、（16）、（17）、（22）和（18），

$$\begin{aligned}
s_2+t_1-t_2+4r_2+4r_1-r_4 &= (3\beta+18)+(60\beta^3+24\beta^2-63\beta+18)-(-60\beta^3-24\beta^2-105\beta+144)\\
&\quad +4(4\beta^3+36\beta^2+3\beta-24)-(-24\beta^3+48\beta^2-18\beta)\\
&= 160\beta^3+144\beta^2+75\beta-204\\
&= 160\left(\frac{3}{4}-h\right)^3+144\left(\frac{3}{4}-h\right)^2+75\left(\frac{3}{4}-h\right)-204\\
&= -160h^3+504h^2-561h+\frac{3}{4} > -160b^3+504a^2-561b+\frac{3}{4}\\
&= \frac{1533890271}{6250000000} > 0.
\end{aligned}$$

根据（13）、（16）、（19）和（22），

$$\begin{aligned}
2s_1-t_1-r_3-3r_1-3r_2 &= 30\beta(2\beta-1)-(60\beta^3+24\beta^2-63\beta+18)\\
&\quad -(20\beta^3-108\beta^2+63\beta+6)-3(4\beta^3+36\beta^2+3\beta-24)\\
&= -92\beta^3+36\beta^2-39\beta+48\\
&= -92\left(\frac{3}{4}-h\right)^3+36\left(\frac{3}{4}-h\right)^2-39\left(\frac{3}{4}-h\right)+48\\
&= 92h^3-171h^2+\frac{561}{4}h+\frac{3}{16} > 92a^3-171b^2+\frac{561}{4}a+\frac{3}{16}\\
&= \frac{18722596069}{62500000000} > 0.
\end{aligned}$$

根据（18）、（13）和（23），

$$\begin{aligned}
2r_4-s_1-r_3-r_2-r_1 &= 2(-24\beta^3+48\beta^2-18\beta)-15\beta(2\beta-1)-(24\beta^3-72\beta^2+66\beta-18)\\
&= -72\beta^3+138\beta^2-87\beta+18\\
&= -72\left(\frac{3}{4}-h\right)^3+138\left(\frac{3}{4}-h\right)^2-87\left(\frac{3}{4}-h\right)+18\\
&= 72h^3-24h^2+\frac{3}{2}h > 72a^3-24b^2+\frac{3}{2}a = \frac{9223413}{7812500000} > 0.
\end{aligned}$$

因此，$t_2+r_4+2r_3 < s_2+t_1+2r_3+4r_2+4r_1 < 2s_1+s_2+r_3+r_2+r_1 < s_1+s_2+2r_4$。

(vii) 根据(14)、(13)、(15)、(17)和(18)，

$$\begin{aligned}
s_0+s_1+s_2+t_2-(1+\beta)(s_1+s_2+2r_4) &= s_0-\beta s_1-\beta s_2+t_2-2(1+\beta)r_4\\
&= (-24\beta^2+48\beta-18)-\beta\cdot 15\beta(2\beta-1)-3\beta(\beta+6)\\
&\quad +\left(-60\beta^3-24\beta^2-105\beta+144\right)\\
&\quad -2(1+\beta)(-24\beta^3+48\beta^2-18\beta)\\
&= 48\beta^4-138\beta^3-96\beta^2-39\beta+126 = 3f(\beta) = 0.
\end{aligned}$$

□

参考文献

[1] Albers S.，《在线调度的更好上界》。*SIAM*计算杂志，29:459-473，1999。[2] Braun O.、Chung F.、Graham R.L.，《四处理器在线调度的下界》。调度杂志，28:529-544，2025。[3] Chen B.、van Vliet A.、Woeginger G.J.，《在线调度的新下界和上界》。运筹学快报，16:221-230，1994。[4] Coffman E.G.、Garey M.R.、Johnson D.S.，《装箱在多处理器调度中的应用》。*SIAM*计算杂志，7:1-17，1978。[5]D´osa G.，《首次适应递减装箱算法的紧上界为 $\mathrm{FFD}(I) \leq \frac{11}{9}\mathrm{OPT}(I) + \frac{6}{9}$》。第一届组合学、算法、概率与实验方法国际研讨会论文集，《计算机科学讲义》，4614:1-11，2007。[6] Faigle U.、Kern W.、Turan G.，《关于划分问题在线算法的性能》。控制论学报，9:107-119，1989。[7] Fleischer R.、Wahl M.，《重新审视在线调度》。调度杂志，3:343-353，2000。[8] Graham R.，《某些多处理异常的上界》。贝尔系统技术杂志，45:1563-1581，1966。[9] Gormley T.、Reingold N.、Torng E.、Westbrook J.，《为请求-回答博弈生成对手》。第*11*届*ACM-SIAM*离散算法研讨会论文集，564-565，2000。[10] Janke M.，《关于四台相同机$\sqrt{3}$ 器上在线最大完工时间最小化的下界》。*arXiv*预印本，arXiv:2611.10440，2026。[11] Johnson D.S.，《近最优装箱算法》。博士学位论文，麻省理工学院，1973。[12] Rudin J.F.，《在线调度问题的改进上界》。博士学位论文，达拉斯德州大学，2001。[13] Rudin J.F.、Chandrasekaran R.，《在线调度问题的改进上界》。*SIAM*计算杂志，32:717-735，2003。[14] Pruhs K.、Sgall J.、Torng E.，《在线调度》。调度手册：算法、模型与性能分析，第15章，CRC出版社，2004。[15] Sleator D.D.、Tarjan R.E.，《列表更新与分页规则的摊销效率》。*Communica-tions of the ACM*，28:202-208，1985。[16] Tan Z.、Zhang A.，《在线与半在线调度》。组合优化手册（第2版），2191–2252，施普林格，2013。[17] Xiao J.、Tan Z.，《在线平行机调度的改进下界（补充材料）》。https://github.com/jwx625/online-makespan-lower-bound-instances