arXiv:2303.01414v1 [cs.DS] 2 Mar 2023

使用压缩和卷积改进单调可塑作业调度算法*

Kilian Grage、Klaus Jansen 和 Felix Ohnesorge

基尔大学，Christian-Albrechts-Platz 4，24118 基尔，德国
{kig,kj}@informatik.uni-kiel.de，felix-eutin@gmx.de

摘要。 在可塑作业调度问题中，需要将一组 n 作业分配给 m 台机器，以最小化处理完所有作业所需的时间。每个作业都是可塑的，因此它不仅可以分配给一台机器，也可以分配给任意数量的相同机器。我们假设每个作业的工作量具有单调性，并且作业可以非连续地放置。本文提出一种（$\frac{3}{2}+\epsilon$）近似算法，其最坏情况运行时间为 $O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\epsilon)+\frac{n}{\epsilon}\log(\frac{1}{\epsilon})\log(\epsilon m))$，适用于 $m \leq 16n$的情况。与已知质量相同的最佳算法相比，该算法在若干对数项上有所改进，因子为 $\frac{1}{\epsilon}$。此外，我们针对机器数量较多的实例 $m > 8\frac{n}{\epsilon}$，提出了一种改进的全多项式时间近似方案，其运行时间为$O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\epsilon))$。由此，当 $m > 16n$时可得到一种运行时间为 $O(n\log^2(\log m))$的$\frac{3}{2}$-近似算法。我们通过一个新的核心观察得到了这些结果：在近似设置下，无需考虑每个作业的所有 m 种可能分配方案。我们将证明，可以将每个作业的相关分配方案数量从 m 减少至 $O(\frac{1}{\epsilon}+\frac{\log(\epsilon m)}{\epsilon})$。利用这一观察即可直接得到改进的全多项式时间近似方案。对于另一项结果，我们首先将问题归约到 Mounié、Rapine 和 Trystram 首次提出的背包问题。我们利用减少后的机器数量设计了一种新的精细舍入方案，并定义了该背包实例的一个修改版本。这使得我们能够应用 Axiotis 和 Tzamos 提出的基于卷积的算法。我们还通过实际实现进一步验证了理论结果，并将我们的算法与此前已知的最佳结果进行了比较。

关键词： 机器调度 · 可变型 · 压缩 · 卷积。

1 引言

机器调度问题是计算机科学中的一个核心问题，其任务是将作业分配给机器，以便在尽可能短的时间内完成所有作业。它的应用不仅限于在一系列处理器核心上执行程序这一常见场景，在现实世界中也有许多应用。例如，可以将机器视为工作人员，将作业视为需要完成的任务或工作。在这种场景下，

* 由德国研究联合会项目 JA 612 /25-1 资助。

多名工作人员可以共同完成一项任务，从而更快地解决问题，这种情况是合理的。然而，这也为该问题增添了一个层面：必须预先为每个作业分配一定数量的机器，并确定其开始时间，由此形成了称为可塑作业并行任务调度的问题。我们的目标是尽可能缩短最后一个作业完成的时间，这一时间称为完工时间。

在这个问题中，处理一个作业所需的时间取决于分配给它的机器数量。本文进一步考虑这样一种情形：作业不仅是可塑的，而且具有单调工作量。作业 j 使用 k 台机器时的工作量定义为 $w(j,\ k) := t(j,\ k) \cdot k$，直观地说，这就是作业所占的面积。我们假设，对于固定的作业 j，该函数随机器数量增加而非递减。这一假设很自然，因为将任务分配到多台机器上并不会减少工作量，反而会因机器间通信而产生一些额外开销。

由于求解该问题的最优解是 NP-hard [11]，我们的目标是提出近似算法。对于每个最优完工时间为 $OPT(I)$ 的实例 I，此类算法都必须保证能找到完工时间至多为 $c \cdot OPT(I)$ 的解，其中 $c > 1$ 是某个乘法近似比。本文介绍两种精度为 $\epsilon > 0$ 的算法：第一种算法保证近似比为 $c_1 = 1 + \epsilon$，运行时间为 $O(n \log^2(\frac{4}{\epsilon} + \log(\epsilon m)\, \epsilon))$，前提是满足附加条件 $m > 8\frac{n}{\epsilon}$。第二种算法的近似比为 $c_2 = \frac{3}{2} + \epsilon$，运行时间为 $O\left(n^2 \frac{1}{\epsilon} + \frac{\epsilon m}{\epsilon} + \right)$，其中满足 $\epsilon = \frac{1}{2}$。如果我们针对 $16n \geq m$ 和 $\frac{n}{\epsilon} \frac{1}{\epsilon} \epsilon m$ 采用第一种算法并结合这两种算法，就能得到一个高效的 $(\frac{3}{2} + \epsilon)$-近似算法。

我们通过一项新的核心观察得出结果：尽管一个作业可以被分配给任意数量的机器，但在寻找近似解时，并非所有 m 种不同的分配方案都可能相关。事实上，我们将证明，如果 m 足够大，就可以将相关的机器分配数量减少至 $O(\frac{1}{\epsilon} + \log(\epsilon m)\ \epsilon)$。这一总体评估基于 Jansen 和 Land [10]提出的压缩概念。

我们利用减少后的相关分配方案数量，通过一个背包问题实例来调度可塑型作业。该方法最初由 Mounié、Rapine 和 Trystram [13]提出。我们给出了一种新的舍入方案，将可塑型作业转换为背包物品，从而定义其背包实例的一个修改版本。我们构造该背包实例，使得不同尺寸和利润的数量较少。这使得能够高效应用 Axiotis 和 Tzamos [1] 利用卷积提出的背包问题算法。该算法在这类实例上表现良好，并且得益于我们的舍入，我们甚至可以在线性时间内高效完成该算法所需的预处理。

1.1 问题定义和记号

本文将围绕两个重要问题展开：第一个是可塑作业并行任务调度，以下简称可塑作业调度。在该问题中，给定一组 J n 作业，以及一组 M m 台相同的机器。对于任意 $l \in \mathbb{N}$，我们记作 $[l] = \{i \in \mathbb{N} \mid 1 \leq i \leq l\}$。

可塑型作业设置下，一个作业的处理时间由函数 $t\colon\ J \times [m] \to \mathbb{R}_{\geq 0}$ 给出，其中 $t(j,k)$表示作业 j 在k 台机器上的处理时间。我们用 $\gamma(j,d)$表示使作业 $=$的处理时间小于$\{i \in [m] \mid t$所需的机器数量，即j,i。如果使用 m 台机器无法达到 d，则称 $\gamma(j,d)$未定义。

对于该问题的一个解，我们需要两项内容：首先是一个分配$\alpha\colon\ J \to [m]$ 以及开始时间的分配 $s\colon\ J \to \mathbb{R}_{\geq 0}$。为简便起见，我们分别记作 $\alpha_j := \alpha(j)$ 和 $s_j = s(j)$。现在，可行解必须满足：在任意时刻，最多有 m 台机器在使用。用$U(t) := \{j \in J \mid t \in [s_j, s_j + t(j, \alpha_j)]\}$ 表示在时间 t处理的作业。如果在所有时刻 $t \in \mathbb{R}_{\geq 0}$ 均有 $\sum_{j \in U(t)} \alpha_j \leq m$，那么由 α 和 s 定义的调度就是可行的。

最后，我们希望最小化该调度的完工时间，即最后一个作业完成的时间。给定分配 α 和开始时间 s，完工时间定义为 $\max_{j \in J}\{s_j + t(j, \alpha_j)\}$。如前所述，一个作业的工作量定义为 $w(j, k) = k \cdot t(j, k)$。在本文中，我们假设每个作业的工作函数都是非递减的。更准确地说，对于所有作业 j和 $k, k' \in [m]$，若 $k \leq k'$，则有 $w(j, k) \leq w(j, k')$。

本研究将讨论的第二个主要问题是背包问题[1]，它也将作为我们求解背包实例的算法的一部分。在背包问题中，给定一组 n 物品，每件物品 i 由一个利润值 $p_i \in \mathbb{R}_{>0}$ 和一个大小或重量 $w_i \in \mathbb{N}$标识。任务是从这些物品中找出一个最大利润子集，使其总重量不超过给定容量 $t \in \mathbb{N}$。

1.2 相关工作

众所周知，即使工作函数单调 [11]，可塑作业调度问题仍然是 NP-hard [7]。此外，除非 P=NP [6]，否则不存在近似比小于$3/2$ 的多项式时间近似算法。Belkhale 和 Banerjee 为具有单调性 [3],的该问题提出了一个 2-近似算法，随后 Turek 等人将其改进到非单调情形 [15]。Ludwig 和 Tiwari 进一步缩短了运行时间 [12]，实现了关于 m的多对数运行时间；这对于紧凑输入编码尤为重要，因为此时输入长度取决于 $\log m$，而非 m。

Mounié 等人提出了一种 $(\frac{3}{2} + \epsilon)$-近似算法，其运行时间为 $O(nm \log \frac{1}{\epsilon})$ [13]。随后，Jansen 和 Land 进一步改进了这一结果：针对具有大量机器的实例提出了一个全多项式时间近似方案，并辅以一种保证 $(\frac{3}{2}+\epsilon)$ 比率且对 m 具有多对数依赖的算法。他们沿用了 Mounié 等人的思路，使用一个背包实例来寻找将作业分配到两个货架上的调度方案，并修改了背包问题以更高效地求解。最近，Wu 等人 [16] 提出了一种新的 $3/2$-近似算法，其运行时间为 $O(nm\log(nm))$ –

[1] 我们主要考虑 $0-1$ 背包问题，不过某些物品可能会出现多次。

背包问题作为子集和的推广，是计算机科学中的另一个同样为 NP-hard 的核心问题。对于这一问题，从 Bellman 的经典动态规划方法开始，人们便研究了运行时间为 $O(nt)$ [4] 的伪多项式算法。近年来，针对最大物品尺寸或不同物品的数量等各种参数，已经取得了许多具有伪多项式运行时间的新成果 [8,14,1,2]。

背包问题与（max $+$）-卷积问题之间出现了一个有趣的联系。在这个问题中，给定两个长度为 $n(a_i)_{0\leq i<n}$，$(b_i)_{0\leq i<n}$ 的序列，需要求出卷积 $c = a \oplus b$，其定义为：对所有 $i \in \mathbb{N}_{<n}$，取 $c_i = \max_{j\leq i}(a_j + b_{i-j})$。该问题可以在二次时间内求解 $O(n^2)$。Cygan 等人 [5] 猜想，可能不存在次二次时间算法，并以此猜想为基础，得出了关于背包问题及类似问题的许多细粒度复杂性结果。Axiotis 和 Tzamos 表明，对于凹序列，可以在线性时间内计算卷积 $O(n)$；他们据此为背包问题提出了一种 $O(Dt)$，其中 D 表示不同物品尺寸的数量 [1]。Polak 等人 [14] 也结合 Eisenbrand Weismantel[8]提出的邻近性论证，运用这一方法为物品尺寸较小的背包问题设计了快速算法。

1.3 我们的结果

我们提出了一种新算法，特别是对于任意精度参数 $\epsilon > 0$，提出了一种 $(\frac{3}{2}+\epsilon)$-近似算法，其运行时间关于 n，$\frac{1}{\epsilon}$ 和 $\log m$ 为多项式。由于我们的算法关于 $\log m$ 为多项式，因此能够处理某些紧凑输入编码，并且通常能够很好地扩展到大型 m。

可塑作业调度的主要难点在于，对于每个作业，我们都需要从 m 种不同的分配方案中进行选择，然后高效地调度作业。不过，我们将证明，并非所有 m 种可能的分配方案都需要考虑。由于我们寻求近似解，且作业具有单调性，因此只需考虑 $O(\frac{1}{\epsilon} \pm \frac{\log(\epsilon m)}{\epsilon})$ 种不同的机器数量。这立即得出一个适用于机器数量较多的实例的全多项式时间近似方案（FPTAS）。

定理 1。设 $\epsilon > 0$。对于满足 $m > 8\frac{n}{\epsilon}$ 的可塑作业调度实例，存在一个$(1+\epsilon)$-近似算法，其运行时间为 $O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\epsilon))$。

若使用 $\epsilon = \frac{1}{2}$，则该结果可用于获得$\frac{3}{2}$-近似算法。

推论 1。考虑满足 $m > 16n$ 的实例上的可塑作业调度。存在一个$\frac{3}{2}$-近似算法，运行时间为 $O(n\log^2(\log m))$。

对于 $m \leq 16n$ 的情况，我们以高效的（$\frac{3}{2}+\epsilon$）近似算法补充这一结果。为此，我们采用与 [10,13] 相同的方法，旨在构造一个背包实例。我们将针对机器数量、处理时间和作业工作量引入一种新的舍入方案，并将这些经过修改的作业转换为背包物品。所得的背包实例中仅包含少量不同的物品尺寸。随后，我们应用 Axiotis 提出的算法

和 Tzamos [1]，该算法在此类实例上表现良好。得益于我们的舍入方案，我们还能够在线性时间内完成其算法的预处理。

定理2。对于可塑作业调度，存在一种算法，对于满足 $m \leq 16n$ 的实例以及任意 $\epsilon > 0$，可在如下时间内得到一个 $\frac{3}{2}+\epsilon$ 近似：$O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\epsilon) + \frac{n}{\epsilon}\log(\frac{1}{\epsilon})\log(\epsilon m))$

这两个结果共同构成了一个（$\frac{3}{2}+\epsilon$）-近似，并在多个方面改进了 Jansen 和 Land [10] 的最佳已知结果。对于较大的 m，我们成功进一步降低了对 m 的依赖。当 m 较小时，我们通过将对 ϵ 的依赖降低 $\frac{1}{\epsilon}$ 倍以及若干个多对数因子，改进了他们的运行时间。我们还认为，与他们的算法相比，我们的算法整体上更为简单，因为我们无需以复杂的方式求解可压缩物品背包问题。相反，我们的算法仅构造修改后的背包实例，并交由 Axiotis 和 Tzamos [1] 提出的一种简单而优雅的算法处理。

结果	Jansen 与 Land [10]	本文
$1+\epsilon$, $(m > 8\frac{n}{\epsilon})$	$O(n\log(m)(\log(m)+\log(\frac{1}{\epsilon})))$	$O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\epsilon))$
$\frac{3}{2}$, $(m > 16n)$	$O(n\log^2(m))$	$O(n\log^2(\log m))$
$\frac{3}{2}+\epsilon$, $(m \leq 16n)$	$O(\frac{n}{\epsilon^2}\log m(\frac{\log m}{\epsilon}+\log^3(\epsilon m)))$	$O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\epsilon) + \frac{n}{\epsilon}\log(\frac{1}{\epsilon})\log(\epsilon m))$

2 多台机器的一般技术与全多项式时间近似方案

本文采用的核心技术是 Jansen 和 Land [10] 引入的压缩 概念。压缩的基本思想是减少分配给作业的机器数量。由于单调性，由此导致的处理时间增加可以得到界定。

引理 1 ([10]). 设 $\rho \in (0, 1/4]$ 为下文所称的压缩因子。现在考虑一个作业 j 和一个机器数量 $k \in \mathbb{N}$，满足 $\frac{1}{\rho} \leq k \leq m$，则有 $t(j, \lfloor(1-\rho)k\rfloor) \leq (1+4\rho)t(j, k)$。

这个引理的直观解释是，如果一个作业使用 $k \geq \frac{1}{\rho}$ 台机器，那么我们最多可以腾出 $\lceil\rho k\rceil$ 台机器，而处理时间会增加 4ρ 倍。下文将通过引入一组预设机器数量来使用这个引理。

定义 1. 令 ρ 为一个压缩因子，并设 $b := \frac{1}{\rho}$。我们将 $S_\rho := \{b] \cup \{\lfloor(1+\rho)^i b\rfloor \mid i \in [\lceil\log_{1+\rho}(\frac{m}{b})\rceil]\}$ 定义为 ρ-压缩尺寸的集合。

注意，将机器数量缩减至 S_ρ 中更小的下一个规模，相当于进行压缩，且处理时间最多只会增加 $1+4\rho$ 倍。

推论 2. 令 $\epsilon \in (0,1)$ 为精度参数，则 $\rho = \frac{\epsilon}{4}$ 是压缩因子，而 $|S_\rho| \in O(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon)$。

通常，我们的算法适用于集合 S_ρ，用于 $\rho = \frac{4}{\epsilon}$，并且只在 S_ρ 中分配机器数量。如果 $m \leq \frac{4}{\epsilon}$，我们会将任意机器数量作为 $S_\rho = [m]$ 处理。我们提出的算法将采用对偶近似框架。

对偶近似框架是调度问题中的经典方法。其基本思想是，针对给定实例，使用近似比为常数 c 的近似算法，从而得到完工时间为 T 的解。虽然这只是一个近似解，但我们可以据此得出结论：最优解的完工时间 T^* 必定落在区间 $[\frac{T}{c}, T]$ 内，并可通过二分搜索在该范围内查找。随后，我们可以将候选项 $d \in [\frac{T}{c}, T]$ 视为对最优完工时间的猜测。

随后，近似算法会与一个估算算法配合使用。该估算算法接收一个实例 I 和一个对完工时间 d 的猜测作为输入。如果存在这样的调度，估算算法必须能够找到完工时间至多为 $(1+\epsilon)d$ 的调度。如果 d 选得太小，即 $(1+\epsilon)d < OPT(I)$，我们的算法就可以拒绝该值 d 并返回假。

我们继续对候选项应用该算法，直到找到 d，使得算法对 d 成功，而对 $\frac{d}{1+\epsilon}$ 不成功。注意，如果算法对 $\frac{d}{1+\epsilon}$ 失败，则有 $d = (1+\epsilon)\frac{d}{1+\epsilon} < OPT(I)$。因此，为 d 生成的解的完工时间为 $(1+\epsilon)d < (1+\epsilon)OPT(I)$。利用二分搜索，我们可以在 $O(\log \frac{1}{\epsilon})$ 次迭代 $[1\underline{0}]$ 内找到这样的候选项 d。

2.1 常数因子近似

我们的常数因子近似分两步进行：首先计算一个分配，并将每个作业分配给若干台机器。其次，使用列表调度来调度现在已固定的并行作业。第一步采用 Ludwig 和 Tiwari [12] 提出的一种算法。

引理 2（[12]）。设有一个用于可塑作业调度的实例 I，其中包含 n 个作业和 m 台机器。对于分配 $\alpha: J \to [m]$，我们用

$$\omega_\alpha := \min\left(\frac{1}{m}\sum_{j\in J} w(j,\alpha(j)), \max_{j\in J} t(j,\alpha(j))\right)$$

表示遵循分配 α的任意调度的显然下界。此外，对于 $S \subseteq [m]$，我们用 $\omega_S :=$ 最小值$_{\alpha: J\to S}\,\omega_\alpha$ 表示可能达到的显然下界，适用于任意分配，其中每个作业都分配给 S 中数量的机器。

对于任意 $S \subseteq [m]$，我们可以在 $O(n$对数$^2\,|S|)$ 时间内计算出一个分配 $\alpha: J \to S$，使得 $\omega_\alpha = \omega_S$。

我们应用这个引理，但将机器数量限制为 $\rho = \frac{\epsilon}{4}$ 的 ρ-压缩尺寸 S_ρ。这样我们得到一个近似值 $\omega_{[m]}$

引理 3。给定一个包含 n 个作业、m 台机器且精度为 $\epsilon < 1$ 的可塑作业调度实例 I。在 $O(n\log^2(\frac{4}{\epsilon} + \log(\epsilon m)\,\epsilon))$ 时间内，我们可以计算出一个分配 $\alpha: J \to [m]$，使得 $\omega_\alpha \leq (1+\epsilon)\,\omega_{[m]}$。

证明。令 $\rho = \frac{\epsilon}{4}, b = \frac{1}{\rho}$ 和 S_ρ 为根据定义 1 得到的 ρ-压缩尺寸集合。现在我们使用引理 2 计算一个分配 α'：$[n] \to S_\rho$，使得 $\omega_{\alpha'} = \omega_{S_\rho}$；注意，所提出的运行时间可由推论 2 和引理 2 得出。剩下要证明的是 $\omega_{\alpha'} \leq (1+\epsilon)\omega_{[m]}$。

为此，令 α 为满足 $\omega_\alpha = \omega_{[m]}$ 的分配。现在，我们通过将其分配的机器数量向下舍入到 S_ρ 中的下一个值来修改该分配。更准确地说，令 α''： $[n] \to S_\rho$; $j \mapsto \max\{s \in S_\rho | s \leq \alpha(j)\}$。注意，根据定义和引理 2，可以立即得到 $\omega_\alpha \leq \omega_{\alpha'} \leq \omega_{\alpha''}$。最后，我们将通过证明 $\omega_{\alpha''} \leq (1+\epsilon)\omega_\alpha$ 来完成证明。

我们注意到，从 α 到 α'' 的舍入是一种压缩。为说明这一点，考虑两个连续的物品尺寸 $\lfloor b(1+\rho)^{(i-1)} \rfloor$, $\lfloor b(1+\rho)^{(i)} \rfloor$，其中 i，并注意：

$$\begin{aligned}\lfloor b(1+\rho)^{(i)} \rfloor - \lfloor b(1+\rho)^{(i-1)} \rfloor &\leq b(1+\rho)^{(i)} - (b(1+\rho)^{(i-1)} - 1) \\ &= b(1+\rho)^{(i)} - b(1+\rho)^{(i-1)} + 1 \\ &= \rho b(1+\rho)^{(i-1)} + 1 \leq \rho b(1+\rho)^{(i)}\end{aligned}$$

由于我们仅在 $\alpha(j) < \lfloor b(1+\rho)^{(i)} \rfloor$ 时向下取整作业，因此可得$\alpha(j) - \alpha''(j) \leq \rho\alpha(j)$。根据引理 1，作业的处理时间最多只会增加至原来的 $1 + 4\rho = 1 + \epsilon$ 倍。因此我们有

$$\max_{j \in J} t(j, \alpha''(j)) \leq \max_{j \in J}\{(1+\epsilon)t(j, \alpha(j))\} = (1+\epsilon)\max_{j \in J} t(j, \alpha(j)).$$

由于工作函数是单调的，因此 $\omega_{\alpha''} \leq (1+\epsilon)\omega_\alpha$ 可直接得出。

□

利用这一分配，我们采用列表调度实现常数因子近似 [9]。我们在双重近似框架中使用这一方法。在接下来的章节中，我们假设已知完工时间的猜测值 d，并给出实现所需结果所需的估计算法。

推论 3。所提出的算法是一种近似算法，其乘法比率为 4，所需时间为 $O(n \log^2(\frac{4}{\epsilon} + \log(\epsilon m)\epsilon))$。

证明。运行时间主要来自应用引理 3，以获得一个满足 $\omega_\alpha \leq (1+\epsilon)\omega_{[m]}$ 的分配 α。对计算得到的分配应用列表调度，可得到完工时间为 $2\omega_\alpha \leq 2(1+\epsilon)\omega_{[m]} \leq 4OPT(I)$ 的调度。□

3 大规模机器数量的全多项式时间近似方案

下文假设，对于每个实例，我们都有 $m > 8\frac{n}{\epsilon}$。Jansen 和 Land 指出，只需在时间 0 将所有作业调度到 j，使用 $\gamma(j, (1+\epsilon)d)$ 台机器上，即可实现全多项式时间近似方案。他们考虑了每项作业所有可能的机器数量。我们认为，只需将数量指定为 $S_{\frac{\epsilon}{4}}$，就能取得类似的结果。不过，我们还需要进行另一次压缩，以确保所得的解可行。

引理4。给定一个包含 n 项作业、 $m > 8\frac{n}{\epsilon}$ 台机器且目标完工时间为 d 的实例 I，我们可以在 $O(n\log(\frac{4}{\epsilon}+\log(\epsilon m)\,\epsilon))$ 时间内，找到完工时间为$(1+3\epsilon)d$的调度（如果 $d \geq OPT(I)$），或者确认 $d < OPT(I)$。

证明。令 S_ρ 为$\rho = \frac{\epsilon}{4}$ 和 $b = \frac{1}{\rho}$ 的 ρ压缩尺寸。令$\gamma'(j,d) := \max\{s \in S_\rho | s \leq \gamma(j,d)$，当 $\gamma'(j,d) \leq b$时，将作业称为窄型作业；当 $\gamma'(j,d) > b$时，则称为宽型作业。我们提出的调度方案是：使用 $\gamma'(j,d)$ 台机器调度窄型作业，并使用压缩数量的机器调度宽型作业，即 $\lfloor(1-\rho)\gamma'(j,d)\rfloor$。我们在时间 0 将所有作业彼此相邻地进行调度。运行时间来自通过二分搜索为所有作业求得 $\gamma'(j,d)$。注意，如果对于某个作业， $\gamma'(j,d)$ 未定义，则说明 d 选得太小。

每个使用 $\gamma(j,d)$ 台机器进行调度的作业 j，其处理时间至多为 d。将机器数向下舍入至 $\gamma'(j,d)$，可能会使处理时间增加 $1+4\rho$ 倍，因为这一过程相当于压缩。随后，我们对宽作业再进行一次压缩，这可能使处理时间再次增加相同的倍数。总之，作业的新处理时间上界为：$(1+4\rho)((1+4\rho)t(j,\gamma(j,d))) \leq (1+\epsilon)^2 d \leq (1+3\epsilon)d$。

接下来需要证明，我们的调度总共使用至多 m 台机器。Jansen 和 Land 证明了 $\sum_{j\in J}\gamma(j,d) \leq m+n$。我们假设 $\sum_{j\in J}\gamma(j,d) > m$，否则我们的调度本身已经可行。用 J_W, J_N 表示宽作业和窄作业的集合。可以看出，$\sum_{j\in J_N}\gamma(j,d) \leq n\cdot b = 4\frac{n}{\epsilon} < \frac{1}{2}m$，因此 $\sum_{j\in J_W}\gamma(j,d) > \frac{1}{2}m$。我们将证明，我们的舍入和压缩过程能够腾出足够的机器。

考虑一个宽型作业 j，并为某个 r 写出 $\gamma(j,d) = \gamma'(j,d)$。由于 j 被分配到 $\lfloor(1-\rho)\gamma'(j,d)\rfloor$ 台机器上，因此释放的机器数至少为：

$$\begin{aligned}\gamma(j,d) - \lfloor(1-\rho)\gamma'(j,d)\rfloor &\geq \gamma'(j,d) + r - (1-\rho)\gamma'(j,d)\\ &= \rho\gamma'(j,d) + r\\ &\geq \rho(\gamma'(j,d)+r) = \rho(\gamma(j,d))\end{aligned}$$

总计我们至少释放 $\sum_{j\in J_W}(\rho\gamma(j,d)) > \rho\frac{1}{2}m > \frac{\epsilon}{4}4\frac{n}{\epsilon} = n$ 台机器。因此，我们的调度最多使用 $\sum_{j\in J}\gamma(j,d) - n \leq m+n-n = m$ 台机器。□

注意，对于 $\epsilon' = \frac{\epsilon}{3}$，我们可以应用这一引理；对于通过向下舍入 $\gamma(j,(1+\epsilon)d)$ 得到的更简化算法，同样可以应用该引理。该算法还允许使用少于 m 台机器的简单调度 [10]。如果我们在对偶近似框架中使用该算法，就能实现所需的全多项式时间近似方案。

证明（定理 *1*）。关于运行时间，我们得出必须应用我们的对偶近似框架，即先应用常数因子近似，然后对于 $\log(\frac{1}{\epsilon})$ 个完工时间猜测应用引理4。结合这些运行时间，我们得到运行时间为 $O(n\log^2(\frac{1}{\epsilon}+\log(\epsilon m)\,\epsilon))$□。

4 ($\frac{3}{2}+\epsilon$)-近似

现在我们将考虑实现 $\frac{3}{2}+\epsilon$ 乘法近似比这一目标。我们的算法将再次在对偶近似框架下运行。因此，我们假设一个完工时间猜测 d并给出一个估算算法。我们的估算算法将以 Mounié 等人 [13]最初提出的方式，将调度问题归约为一个背包实例。Jansen 和 Land [10] 也采用了这种方法，并给出了该背包实例的一个修改版本。然而，我们提出了一种新的、更简单的舍入方案，该方案对 $\rho=\frac{4}{\epsilon}$ 使用 ρ压缩尺寸，并进一步修改物品利润。这样，我们无需使用复杂的算法来求解背包问题，而是可以高效地应用 Axiotis 和 Tzamos [1]的结果。

一开始，我们将作业集划分为小作业和大作业 $J=J_B(d)\cup J_S(d)$，其中 $J_S(d):=\{j\in J\mid t(j,1)\leq\frac{d}{2}\}$ 以及 $J_B(d)=J\backslash J_S(d)$。由于我们可以在线性时间内在末尾贪心地添加小物品 [10], ，因此只需调度大作业。下面简要介绍 Mounié 等人提出的背包实例中最重要的结果。

他们的主要思路是将所有作业分配到两个宽度为 m 的货架上。第一个货架 S_1 的高度为 d ，第二个货架 S_2 的高度为 $\frac{d}{2}$ 。如果作业j 被调度到高度为 $s\in\{d,\frac{d}{2}\}$ 的任一货架上，那么 j 将被分配到 $\bar{\gamma}(j,s)$ 台机器上。为了将作业分配到货架，他们使用以下背包实例：

对于每个作业 $j\in J_B(d)$，考虑一个大小为 $s_j(d):=\gamma(j,d)$、利润为$p_j(d):=w(j,\gamma(j,d/2))-w(j,\gamma(j,d))$ 的物品，并将背包容量设为 $t:=m$。直观而言，这个背包实例会选出一组作业 J' ，以便在 S_1 中进行调度。选出的这些作业使得它们在 S_2 中的工作量增幅较大。

我们将此问题记为 $KP(J_B(d),m,d)$，其中前两个参数定义物品和背包容量，第三个参数是目标完工时间，它进而决定物品的大小和利润。给定一个解 $J'\subseteq J$ ，我们用$W(J',d)$ 表示由此得到的双层架调度的总工作量，并注意到：

$$\begin{aligned}W(J',d)&=\sum_{j\in J'}w(j,\gamma(j,d))+\sum_{j\in J_B(d)\backslash J'}w(j,\gamma(j,\frac{d}{2}))\\&=\sum_{j\in J_B(d)}w(j,\gamma(j,\frac{d}{2}))+\sum_{j\in J'}w(j,\gamma(j,d))-\sum_{j\in J'}w(j,\gamma(j,\frac{d}{2}))\\&=\sum_{j\in J_B(d)}w(j,\gamma(j,\frac{d}{2}))-\sum_{j\in J'}p_j(d)\end{aligned}$$

由于背包利润最大化，总工作量 $W(J',d)$ 最小化。我们所采用的 Mounié 等人的结果总结在以下两个引理中。关于这些结果的详细说明，请参见 [10,13] 。

引理 5 ([13]). 如果存在完工时间为 d 的调度，那么就存在一个解$J'\subseteq J_B(d)$，用于背包实例 $W(J',d)\leq md-W(J_S(d),d)$。

引理 6 ([13]). 如果背包实例 $W(J', d) \leq md - W(J_S(d)\ d)$存在一个解 $J' \subseteq J_B(d)$，那么我们可以在时间 $O(n \log n)$ 内，为所有作业 J 找到一个完工时间为 $\frac{3}{2}d$ 的调度。

基于这些引理，如果 $W(J',\ d)$ 大于 $md - W(J_S(d)\ d)$，我们可以轻易地拒绝完工时间猜测 d。我们还注意到，如果我们为更大的完工时间找到一个解，则可以应用引理6。

推论 4（[10]）。设 $d' \geq d$ 和 $J' \subseteq J_B$（d）是背包问题 KP（J_B（d） m，d'）的一个可行解，其中 W（J'，d） $\leq md' - W$（J_S（d） d）。那么我们可以在时间 O（$n \log n$）内找到一个完工时间至多为 $\frac{3}{2}d'$ 的调度。

现在，我们构造一个修改后的背包实例，以便对 $d' = (1+4\epsilon)d$ 应用该推论。首先，我们将 $\rho = \frac{\epsilon}{4}$ 的机器数量缩减为 ρ-压缩尺寸。考虑 S_ρ 和 $b := \frac{1}{\rho}$，并令 $\gamma'(j, s) := \max\{k \in S_\rho | k \leq \gamma(j, s)\}$，其中 j 为任意作业，$s \in \{\frac{d}{2}, d\}$。用 $\widetilde{p_j}(d) := \gamma'(j, \frac{d}{2})t(j, \gamma'(j, \frac{d}{2})) - \gamma'(j, d)t(j, \gamma'(j, d))$ 表示随后将进一步修改的中间利润。

如果一个作业在相应货架中使用的机器数超过 b，即对于相应的 $s \in \{\frac{d}{2}, d\}$，满足 $\gamma'(j, s) \geq b$，我们进一步称该作业在该货架中是宽型 作业。如果作业不是宽型作业，则对于某个货架，我们称其为窄型 作业。

对于在两个货架中均为窄型的作业，我们将直接修改其利润。设 j为一个作业，对于两个$s \in \{\frac{d}{2}, d\}$均满足 $\gamma'(j, s) < b$，则我们通过将中间利润向上取整到 ϵd 的下一个倍数来修改它，令$p'_j(d) := \min\{i\epsilon d \,|\, i\epsilon d \geq \widetilde{p_j}(d) \text{ and } i \in \mathbb{N}^*_{\leq \frac{2}{\epsilon^2}}\}$。这是有定义的，因为在这种情况下，原始利润受 $w(j, \frac{d}{2}) < b\frac{d}{2} = \frac{2}{\epsilon^2}\epsilon d$的界。为便于后续论证，记修改后的工作量为 $w'(j, \frac{d}{2}) := w(j, \frac{d}{2})$，并令$w'(j, d) := w'(j, \frac{d}{2}) - p'_j(d)$。

对于在两个货架上都是宽型的作业 j，也就是说，当 $\gamma'(j, \frac{d}{2}) \geq \gamma'(j, d) \geq b$ 时，我们将修改处理时间。具体而言，我们设定 $t'(j, s) := \frac{1}{1+4\rho}s$，其中 $s \in \{\frac{d}{2}, d\}$，由此得到修改后的工作量值 $w'(j, s) := t'(j, s)\gamma'(j, s)$。然后，我们根据修改后的工作量定义新的利润：$p'_j(d) := w'(j, \frac{d}{2}) - w'(j, d)$。

剩下的是在一个货架上为窄型、在另一个货架上为宽型的作业。考虑这样的作业 j，其 $\gamma'(j, \frac{d}{2}) \geq b > \gamma'(j, d)$。对于窄型版本，我们再次对处理时间 $t'(j, \frac{d}{2}) := \frac{1}{1+4\rho}\frac{d}{2}$进行舍入，得到 $\overline{w'(j, \frac{d}{2})} := \overline{t'(j, \frac{d}{2})}\gamma'(j, \frac{d}{2})$。对于宽型作业，我们将工作量 $w(j, \gamma'(j, d))$ 向下舍入到下一个 $i\epsilon d$ 的倍数。具体地说，我们设定 $w'(j, d) := \max\{i\epsilon d \,|\, i\epsilon d \leq w(j, \gamma'(j, d))$ 和 $i \in \mathbb{N}_{\leq \frac{4}{\epsilon^2}}\}$。注意，未修改的工作量有界于 $w(j, \frac{d}{2}) \leq w(j, d) < bd = \frac{4}{\epsilon}d = \frac{4}{\epsilon^2}\epsilon d$。然后得到修改后的利润值 $p'_j(d) = w'(j, \frac{d}{2}) - w'(j, d)$。

使用这些修改后的利润和尺寸 $s'_j(d) = \gamma'(j, d)$，然后我们求解由此得到的问题 $KP'(J_B(d)\ m,\ d,\ \rho)$，以获得一个最优物品集 J'。

引理 7. 令 J' 为 $KP'(J_B(d)\ m, d, \rho)$ 和 $d' = (1 + 4\epsilon)d$ 的一个解，则在处理时间和机器数量不变的情况下，J' 也是 $KP(J_B(d)\ m, d')$ 的一个解。此外，若存在完工时间为 d 的调度，则有 $W(J', d') \leq md' - W(J_S(d)\ d)$。

证明. 对于第一部分，我们需要证明，当每个货架允许的处理时间为 d' 或 $\frac{d'}{2}$ 时，J' 中的所有作业都能装入相应的背包。考虑所有满足 $\gamma(j,d) \leq b$ 的作业 $j \in J'$，并注意到这些作业在两个背包问题实例中的大小相同，因为 $\gamma(j,d') \leq \gamma(j,d)$。对于任意宽作业 $j \in J'$，有 $t(j,\gamma'(j,d)) \leq (1+4\rho)d \leq d'$，因此 $\gamma(j,d') \leq \gamma'(j,d)$。由此得到 $\sum_{j\in J'} \gamma(j,d') \leq \sum_{j\in J'} \gamma'(j,d) \leq m$，因为 J' 求解了容量为 m 的修改后的背包实例。

在考虑 J' 的总工作量之前，我们先对舍入作一些观察：通过舍入尺寸，我们减少了每个作业可用机器的数量。由于单调性，与 Mounié 等人的原始背包实例相比，这只会减少每个作业的工作量。随后，我们进一步减少窄作业的工作量，减少幅度至多为 ϵd；并将宽作业的处理时间缩短 $\frac{1}{1+4\rho}$ 倍。

注意，对于宽型作业 j 和货架大小 s，设定 $t'(j,s) = \frac{1}{1+4\rho}s$ 实际上是在缩短处理时间，这一点可以通过间接证明看出。因此，假设 $t(j,\gamma'(j,s)) < \frac{1}{1+4\rho}s$，并令 $s_{k+1} := \gamma'(j,d)$，再令 s_k 为 S_ρ 中下一个较小的尺寸。将机器数量减少到 s_k 属于压缩，因此我们有 $t(j,s_k) \leq (1+4\rho)t(j,s_{k+1}) < s$。由此可见，$\gamma'(j,s)$ 并非取最小值。

一般而言，我们有 $w'(j,s) \leq w(j,s)$，并希望继续给出 $w(j,s)$ 的一个上界。注意，我们可以假设处理时间不会随着机器数量的增加而增加。否则，我们只需省略那些会增加处理时间的机器数量，并始终安排在较少数量的机器上。由此可得 $w(j,s) \leq (1+\rho)\gamma'(j,s)t(j,\gamma'(j,s))$。

注意，对于货架 2 中的作业，我们只会缩短宽型作业的处理时间，因此可得：

$$w(j,\frac{d}{2}) \leq (1+\rho)\gamma'(j,\frac{d}{2})(1+4\rho)t'(j,\gamma'(j,\frac{d}{2})) = (1+\rho)(1+4\rho)w'(j,\frac{d}{2}).$$

对于货架 1 中的宽作业，我们也采用相同的做法。不过，对于该货架中的窄作业，我们会进一步将工作量减少 ϵd。对 $w(j,d)$ 进行与 $w(j,\frac{d}{2})$ 相同的估算，并加上这部分额外增量，我们可以得出：$w(j,d) \leq (1+\rho)(1+4\rho)w'(j,d) + \epsilon d$。

对于命题的第二部分，由引理 5 可知，存在一个最优解 J^*，满足 $KP(J_B(d), m, d)$，且 $W(J^*, d) \leq md - W(J_S(d), d)$。此外，$J^*$ 也是修改后的背包问题的一个可行解，因为我们的修改只会减小物品尺寸。与原实例类似，我们的修改后的背包实例将最大化背包收益，进而使修改后的工作量值下的双层架调度的总工作量最小化。由于 J' 是修改后实例的最优解，因此 J' 的修改后总工作量大于 J^* 的修改后工作量。具体而言，我们有：

$$\sum_{j\in J'} w'(j,d) + \sum_{j\in J_B(d)\setminus J^*} w'(j,\frac{d}{2})) \leq \sum_{j\in J^*} w'(j,d) + \sum_{j\in J_B(d)\setminus J^*} w'(j,\frac{d}{2})).$$

现在我们可以得出结论：由 J' 所蕴含的双层架调度的总工作量有界：

$$
\begin{aligned}
W(J',d) &= \sum_{j\in J'} w(j,d) + \sum_{j\in J_B(d)\setminus J'} w(j,\frac{d}{2}) \\
&\leq \sum_{j\in J'} ((1+\rho)(1+4\rho)w'(j,d)+\epsilon d) + \sum_{j\in J_B(d)\setminus J'} (1+\rho)(1+4\rho)w'(j,\frac{d}{2}) \\
&\leq |J'|\epsilon d + (1+\rho)(1+4\rho)(\sum_{j\in J'} w'(j,d) + \sum_{j\in J_B(d)\setminus J'} w'(j,\frac{d}{2})) \\
&\leq |J'|\epsilon d + (1+\rho)(1+4\rho)(\sum_{j\in J^*} w'(j,d) + \sum_{j\in J_B(d)\setminus J^*} w'(j,\frac{d}{2})) \\
&\leq |J'|\epsilon d + (1+\rho)(1+4\rho)(\sum_{j\in J^*} w(j,d) + \sum_{j\in J_B(d)\setminus J^*} w(j,\frac{d}{2})) \\
&\leq m\epsilon d + (1+\rho)(1+4\rho)(md - W(J_S(d),d)) \\
&\leq (1+4\epsilon)md - W(J_S(d),d) = md' - W(J_S(d),d)
\end{aligned}
$$

最后，由于工作量具有单调性，我们还可得 $W(J',d') \leq W(J',d)$，由此完成证明。

□

4.1 求解背包问题

如前所述，我们打算使用 Axiotis 和 Tzamos 的一种算法[1]。该算法主要分为两个步骤。第一步，将背包实例中的物品划分为若干集合，每个集合中的物品尺寸相同。然后分别针对每个物品集合求解背包问题。对于每种物品尺寸 s 及其对应的物品集合 $I_s = \{i \in I \mid s_i = s\}$，都会生成一个解数组 R_s，其中$R_s[t']$ 表示仅使用尺寸为 s的物品时，容量为 $t' \leq t$ 的背包所能获得的最大利润。请注意，根据该问题的性质， $R_s[t']$ 始终等于 I_s中利润最高的$\lfloor \frac{t'}{s} \rfloor$ 个物品的利润总和。

这些解数组 R_s 具有如下特殊结构：对于所有 $s' < s$ 和 $k \in \mathbb{N}$, $R_s[k\cdot s] = R_s[k\cdot s + s']$。进一步考虑唯一条目，我们有$R_s[(k+1)\cdot s] - R_s[k\cdot s] \geq R_s[(k+2)\cdot s] - R_s[(k+1)\cdot s]$，对于每个 k，因为所添加物品的利润逐渐减少。这种结构也称为 s-步凹序列，因为唯一条目构成了一个凹序列。在算法的第二步中，他们按照顺序通过卷积组合这些解数组，以生成最终的解数组 $R = R_1 \oplus R_2 \oplus \cdots \oplus R_{[s_{max}]}$。

Axiotis 和 Tzamos 的一个非常重要的结果是，如果这些卷积按顺序进行，那么对于某个相应的 s，其中一个序列始终会是 s-凹的。他们在论文中证明，与已知的最佳二次时间相比，与一个 s-步长凹序列进行卷积可以在线性时间内完成。

引理 8（[1]）。给定任意序列 A 和 R_h，对于某个 $h \in \mathbb{N}$，且每个序列都有 t 个元素，我们可以在时间 $O(t)$ 内计算卷积 $A \oplus R_h$。

在我们的设定中，背包容量为 $t = m$。得益于舍入，我们只有 $|S_\rho|$ 种不同的物品尺寸，这决定了需要计算的卷积次数。不过，我们还必须计算由每种尺寸中利润最高的物品组成的初始解。得益于对物品利润进行舍入，我们还可以高效地对这些物品排序，从而生成初始解数组 R_h。

引理 9。给定一个修改后的背包实例 $KP'(J_B(d)\ m, d, \rho)$，我们可以在时间 $O(n + m(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon))$ 内计算所有 $t \le t$ 的条目 $R_h[t']$。

证明。我们的目标是按利润对物品排序，然后将利润最高的物品依次加入，以填充数组 R_h。我们将根据物品的舍入方式对其进行排序：

考虑具有 $\gamma'(j, s) < b$ 的作业 j，它们同时满足 $s \in \{\frac{d}{2}, d\}$，并用 n_1 表示这类作业的数量。将它们的利润按 $\frac{1}{\epsilon}\frac{1}{d}$ 缩放后，可得到形式为 $\tilde{p}_j(d) = i$ 的利润，其中某个 $i \in \mathbb{N}_{\le \frac{2}{\epsilon^2}}$。我们可以使用基数排序对利润进行排序，所需时间为 $O(n_1 + \frac{1}{\epsilon})$；对利润进行编码时使用 $O(1)$ 位数字，取值范围为 0 到 $\frac{1}{\epsilon}$。

现在考虑具有 $\gamma'(j, \frac{d}{2}) \ge \gamma'(j, d) \ge b$ 的 n_2 作业 j。若将这些物品的利润按 $\frac{1+4\rho}{d}$ 缩放，则有 $\tilde{p}_j(d) = \frac{1}{2}\gamma'(j, \frac{d}{2}) = \gamma'(j, d)$。这些物品可以通过桶排序按利润排序，所需时间为 $O(n_2 + m)$。

对于剩余的 n_3 个作业 j（其中 $\gamma'(j, \frac{d}{2}) \ge b > \gamma'(j, d)$），我们需要考虑修改后的利润 $p'_j(d) := 2^{\frac{d}{(1+4\rho)}\gamma'(j, \frac{d}{2}) - i\epsilon d}$，其中某个 $i \in \mathbb{N}$。我们将这些利润按 $2^{(1+\epsilon)\, d\epsilon^2}$ 进行缩放，得到 $\tilde{p}_j(d) = \gamma'(j, \frac{d}{2})\frac{1}{\epsilon^2} = \frac{2id}{\epsilon} \le \frac{m}{\epsilon^2}$。通过将利润编码为两个取值范围为 0 到 $\frac{m}{\epsilon}$ 的数字，可以在 $O(n_3 + \frac{m}{\epsilon})$ 时间内使用基数排序对这些物品进行排序。

将这三个步骤合起来需要 $O(n_1 + n_2 + n_3 + \frac{1}{\epsilon} + m + \frac{m}{\epsilon}) = O(n + \frac{m}{\epsilon})$ 时间。此外，我们可以用归并排序在 $O(n)$ 时间内合并这三个已排序列表，并遍历所有物品以填充实际的解数组。需要填充的条目总数至多为 $m(\frac{4}{\epsilon} + \log(\epsilon m)\,\epsilon)$，因为每个数组中有 m 个条目，并且每种物品尺寸都有一个数组。□

严格来说，我们只需要这些解数组中的唯一条目即可应用算法 [14]。这些条目实际上可以在 $O(n + \frac{m}{\epsilon})$ 时间内计算出来，但无论如何，合并所有数组都会主导运行时间。

推论5。我们可以在 $O(m(|S_\rho|))$ 时间内计算 $R_1 \oplus R_2 \oplus \cdots \oplus R_{|S_\rho|}$。

利用这个背包问题解，我们可以根据推论 4 构造一个调度。值得注意的是，使用 Mounié 等人 [13]提出的过程进行的这一最终构造，可以通过使用舍入后的处理时间[10]，在 $O(n)$ 时间内实现。

证明（定理*2*）。

我们采用对偶近似框架，这意味着我们以 $O(n\log^2(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon))$ 的时间为 d 计算一个上界。最终，我们得到 $\log(\frac{1}{\epsilon})$ 个 d 的候选项，并为每个候选项构造背包实例。

为此，我们需要确定它们在压缩尺寸中的机器数量。通过二分搜索，可以在 $O(n \log(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon))$ 时间内完成。对背包物品的所有进一步修改都可以在 $O(n)$ 时间内完成。对于所有候选项，这些步骤总共需要 $O(n \log^2(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon))$ 时间。

求解每个候选项对应的结果背包问题，运行时间为$O(m(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon)) \subseteq O(m\frac{1}{\epsilon}\log(\epsilon \bar{m}))$。将此方法应用于所有候选项，并且由于$m \leq 16n$，最终运行时间为 $O(n \log^2(\frac{1}{\epsilon} + \log(\epsilon m)\,\epsilon) + \frac{n}{\epsilon}\log(\frac{1}{\epsilon})\log(\epsilon m))$。

□

5 实现

我们实现了本文介绍和使用的所有算法，以及 Jansen 和 Land 提出的算法的一个版本 [10]。我们注意到，由于他们用于求解可压缩物品背包问题的算法十分复杂精巧，因此我们没有实现其最终版本。相反，我们的实现会计算他们修改后的背包实例，并通过他们提出的动态规划方法求解。

实现和实验均在 Raspberry Pi 4 Model B 上进行。由于没有采用任何并行化方式，我们将实验限制为使用一个 CPU 核心。我们已将实现的一个版本上传至 GitHub（https://github.com/Felioh/MoldableJobScheduling）。接下来，我们主要测试了 $m \leq 16n$ 部分，因为我们认为这是对两种结果进行比较时更有参考价值的部分。

5.1 计算结果

在测试实例方面，我们为可塑作业调度生成了多组随机实例。机器数量主要在 30 到 100 之间，作业数量则在 10 到 120 之间。我们针对 $\epsilon = \frac{1}{10}$ 对这些实例进行了测试，结果见附录中的图表。图 1 和图 2 展示了我们的算法与 Jansen 和 Land 的算法之间的平均运行时间差异。请注意，此处的差值计算方式是用他们算法的运行时间减去我们算法的运行时间。因此可以看出，对于所分析的作业数量和机器数量，我们的算法略胜一筹，而且随着机器数量和作业数量增加，我们的算法似乎具有更好的扩展性。

在图 3 至图 5 中，我们比较了两种算法的平均完工时间，以评估解的质量。在大多数情况下，两种算法的解的质量总体上相当，但有时我们的算法略胜一筹。我们认为，我们的算法在解的质量方面表现更好，是因为采用了舍入。首先，将机器数量舍入到 S_ρ 中的取值，本质上是一种压缩，但并未充分利用引理 1 带来的潜力。由于我们没有尽可能大幅减少机器数量，实际误差更小。类似地，对背包物品所做的其他调整，主要是为了形成简单的结构，同时也控制额外误差。

6 结论

本文介绍了我们提出的新型 $\frac{3}{2}+\epsilon$-近似算法，它结合了可塑调度、背包问题和卷积中的多种技术。我们的算法在理论上改进了该问题已知的上界，而且从实验结果来看，在实践中也更快。我们的结果带来的一项有趣启示是，借助压缩，只需将可塑调度简化到特定的一组机器数量即可。事实上，若要寻找一个近似解，就没有必要考虑所有可能的分配方案。

参考文献

1. Axiotis, K., Tzamos, C.：《容量约束动态规划：更快的背包与图算法》。载于：第46届自动机、语言与编程国际研讨会（ICALP 2019）。（LIPIcs），第132卷，第19:1–19:13页。德国达格施图尔（2019） 2. Bateni, M., Hajiaghayi, M., Seddighin, S., Stein, C.：通过卷积与预测实现的快速背包算法。载于：第50届ACM SIGACT计算理论年度研讨会论文集。第1269–1282页。STOC 2018，美国纽约州纽约市（2018） 3. Belkhale, K.P., Banerjee, P.：可划分独立任务调度问题的近似算法。载于：国际并行处理会议（ICPP）。第72–75页（1990） 4. Bellman, R.：动态规划。载于：普林斯顿大学出版社（1957） 5. Cygan, M., Mucha, M., Wegrzycki, K., Wlodarczyk, M.：关于等价于（min,+）-卷积的问题。载于：第44届自动机、语言与编程国际研讨会（ICALP 2017）。（LIPIcs），第80卷，第22:1–22:15页（2017） 6. Drozdowski, M.：多处理器任务调度的复杂性。《波兰科学院技术科学通报》**43**，381–392（1995） 7. Du, J., Leung, J.Y.T.：并行任务系统调度的复杂性。《SIAM离散数学杂志》**2**(4)，473–487（1989） 8. Eisenbrand, F., Weismantel, R.：使用Steinitz引理的整数规划邻近性结果与更快算法。《ACM算法汇刊》**16**(1)（2019年11月） 9. Garey, M.R., Graham, R.L.：带资源约束的多处理器调度界。《SIAM计算杂志》**4**，187–200（1975） 10. Jansen, K., Land, F.：线性时间调度单调可塑型作业。载于：2018年IEEE国际并行与分布式处理研讨会（IPDPS）。第172–181页。IEEE计算机学会，洛斯阿拉米托斯，美国（2018年5月） 11. Jansen, K., Land, F., Land, K.：《调度与装箱问题算法运行时间的界》，计算机科学研究所报告，第1302卷（2013） 12. Ludwig, W., Tiwari, P.：可伸缩与不可伸缩并行任务的调度。载于：第五届ACM-SIAM离散算法年度研讨会论文集。第167–176页。SODA '94，工业与应用数学学会，美国（1994） 13. Mounié, G., Rapine, C., Trystram, D.：独立单调可伸缩任务调度的3/2对偶近似。《SIAM计算杂志》**37**，401–412（2007年1月）

14. Polak, A., Rohwedder, L., Węgrzycki, K.：《小物品背包与子集和》。载于：第48届自动机、语言与编程国际研讨会（ICALP 2021）。（LIPIcs），第198卷，第106:1–106:19页。德国达格施图尔（2021） 15. Turek, J., Wolf, J.L., Yu, P.S.：并行化任务调度的近似算法。载于：第四届ACM并行算法与体系结构研讨会论文集。第323–332页。SPAA '92，计算机协会，美国纽约州纽约市（1992） 16. Wu, F., Zhang, X., Chen, B.：单调可塑任务调度的改进近似算法。《欧洲运筹学杂志》**306**(2)，567–578（2023）。https://doi.org/https://doi.org/10.1016/j.ejor.2022.08.034, https://www.sciencedirect.com/science/article/pii/S0377221722006762

A 计算结果（图表）

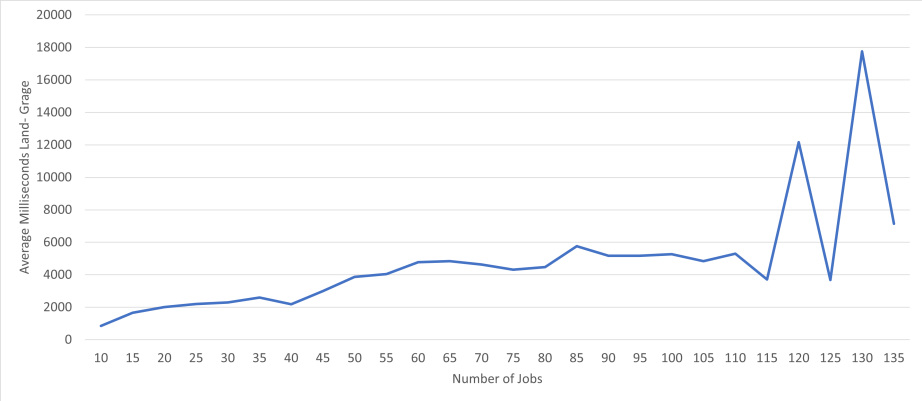

图 1. 与作业数量相关的平均运行时间差。

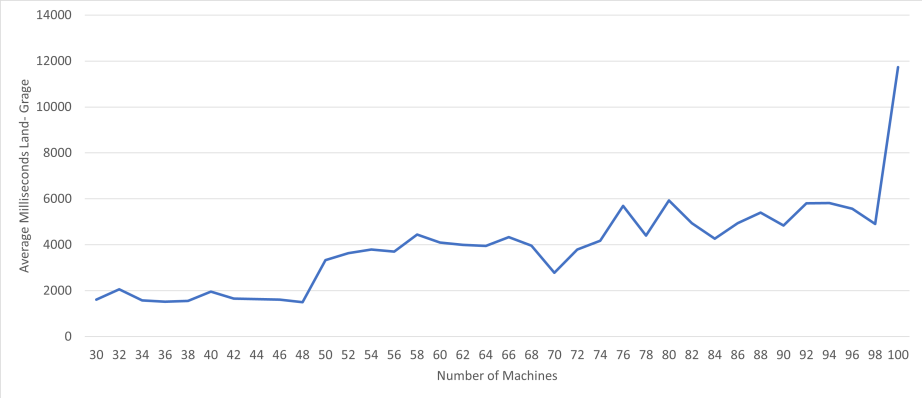

图 2. 与机器数量相关的平均运行时间差。

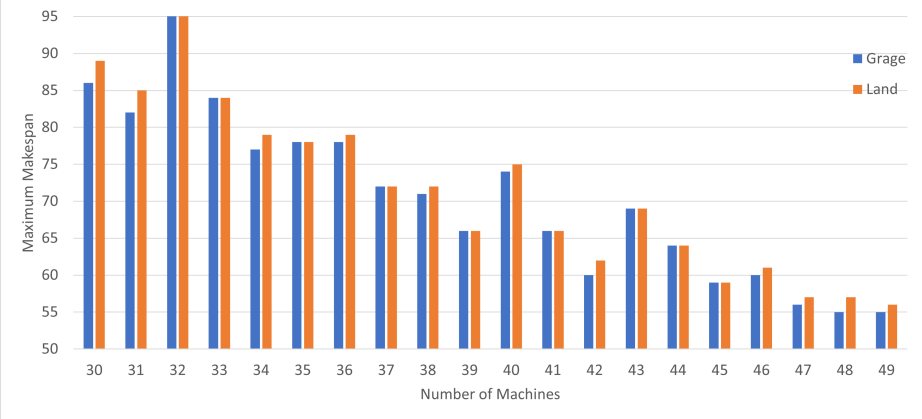

图3。 平均总工期比较，仅限机器数量相同的实例。

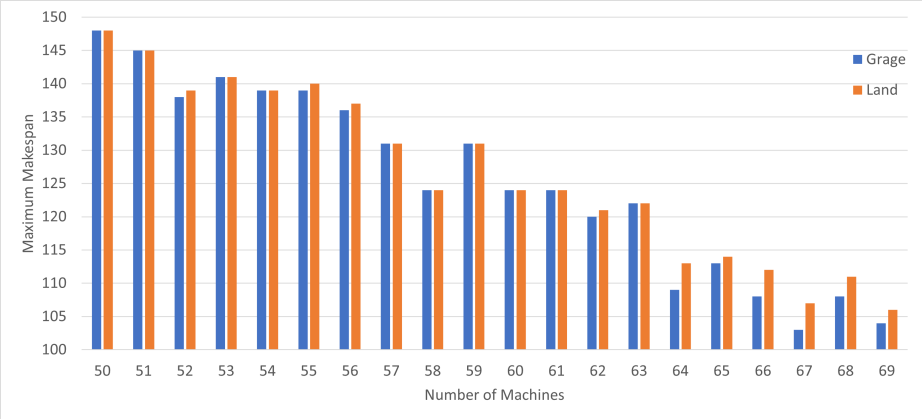

图4。 平均总工期比较，仅限机器数量相同的实例。

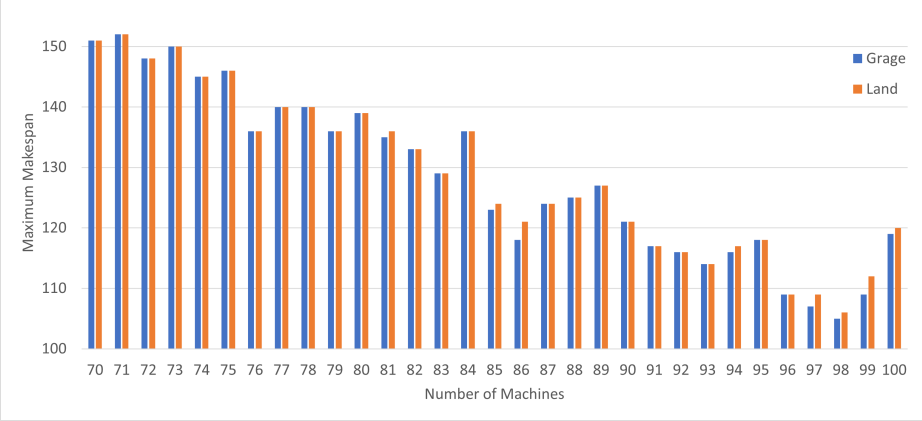

图 5. 仅比较机器数量相同的实例的平均总工期。