arXiv:2602.11366v1 [math.CO] 11 Feb 2026

积木堆叠、飞机加油与鲁棒预约调度

Simon Gmeiner 和 Andreas S. Schulz

摘要

如何摆放一叠相同木块，使其尽可能远地悬伸出桌子边缘？我们将这道经典谜题推广到宽度和质量各不相同的木块。尽管这个问题乍看之下十分简单，我们证明，要高效地确定最大悬伸量的堆叠配置并非易事。形式上，我们证明积木堆叠问题是 NP 困难的，从而部分回答了文献中的一个开放问题。此外，我们还发现，不使用配重块的叠堆与另一道著名谜题——飞机加油问题，以及一个具有实际意义的问题——鲁棒预约调度之间存在出人意料的联系。除了揭示它们与现实世界中如何在不确定条件下制定日程安排这一挑战之间的显著关系外，这种等价性还为无平衡配重的积木堆叠问题提供了一个多项式时间近似方案，即 $(1+\epsilon)$-近似算法；对于一般情形，则提供了一个 $(2+\epsilon)$-近似算法。

1 引言。

堆叠一组块时，能够实现的最大悬伸量是多少？这是积木堆叠问题的核心问题。这个谜题以多种形式流传了数十年（参见例如 [18] 可追溯至 19 世纪中叶的参考文献）。通常，问题假定所有块都完全相同，且只能一个接一个地上下堆叠。值得注意的是，即使在这种限制重重的形式下，理论上也可以构造出悬伸量任意大的平衡堆叠。因此，这个问题引起了广泛关注。它已成为广为流传的谜题和实验，出现在许多教科书以及互联网上数不胜数的视频中，尽管名称各不相同且使用并不一致，例如“Leaning Tower of Lire”（如 [11]）、“积木堆叠问题”或“书籍堆叠问题”、“建造无限长桥”等。除了块和书籍，人们也曾使用砖块、硬币或扑克牌等其他物品来搭建这类叠堆。然而，其基本原理始终相同。

众所周知（例如参见 [7, 11, 18]），对于 n 个宽度为二的相同木块，通过反复放置，使最上方 i 个块的重心恰好位于下方

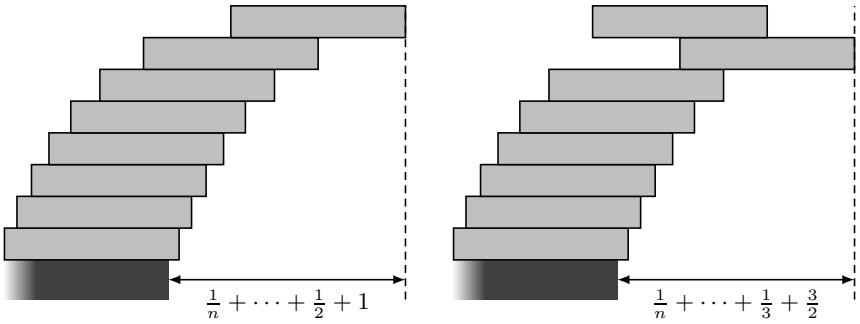


图1：对于宽度为二的相同木块，调和堆叠与可实现最大悬伸量的另一种平衡堆叠构型。

木块的边缘（对所有 $i = 1, \ldots, n$均如此），便可得到悬伸量为 $\sum_{k=1}^{\infty} 1/k$的平衡堆叠。由于调和级数 $\sum_{k=1}^{\infty} 1/k$发散，图1左侧所示的这种所谓调和堆叠，其悬伸量对于足够大的 n可以任意大。尽管人们早已推测这种构造方法能达到可能的最大悬伸量，但直到最近，Treeby [24] 才给出了严谨且确凿的证明。然而，正如Hall [7], 早已指出的那样，这一解法或许令人惊讶地并非唯一：通过让倒数第二层的块伸出，同时将最上层积木用作平衡配重，也能达到相同的悬伸量（参见图1）。Hall [7] 进一步探究了平衡配重的概念，并发现每层允许放置多个块，可以在块数相同的情况下实现更大的悬伸量；Paterson等人 [18, 19] 后来对这一变体进行了详细研究。其他改进方案也得到了探索，例如采用重量分布不均匀的块 [10, 12], 、黏性木块 [14], 或连续版本 [21], 。

Treeby [24] 率先将积木堆叠问题推广到由一组不同木块组成的情形。他仍要求所有木块高度相同，但允许它们的宽度不同。更具体地说，他假设木块的质量与其宽度成正比。在这种情况下，木块不再可以互换，因此堆叠顺序会影响可实现的悬伸量。如果不允许使用平衡配重，只需按木块宽度排序，并将最宽的木块放在顶部，即可得到最优堆叠顺序（见图 2）。然而，与相同木块的情形不同，配重块可以显著增加悬伸量。因此，要找到最大可能悬伸量就困难得多，因为必须考虑复杂得多的堆叠配置。Treeby甚至猜想，找到最优解是 NP 困难的。有关NP 困难性和计算复杂性的概述，请参见第3节。

我们比 Treeby 更进一步，考虑块之间在高度、材料或三维深度方面的差异。由于只有相对于横截面的重心才能决定堆叠是否平衡，这些因素都不会直接影响堆叠的悬伸量。

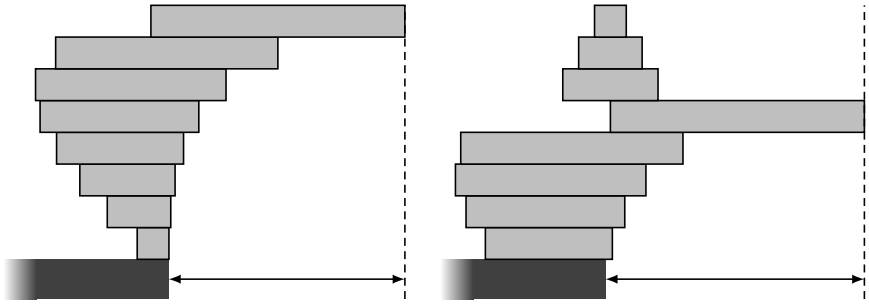


图 2：不使用平衡配重时，将高度相同的块按宽度从上到下非递增排列，可获得最大悬伸量。将部分块用作配重，则能获得更大的悬伸量。

相反，它们的影响可以汇总到单个块的质量中，而这又会对寻找理想的堆叠配置产生巨大影响。

如图 2 所示，我们通常希望最宽的块突出。不过，如果该块的质量明显过大，将它放在堆叠中更靠下的位置可能更好，如图 3 所示。相比之下，短而重的块理想情况下应放在堆叠底部附近，或用作配重，以增加突出块的悬伸量。积木堆叠问题旨在找出能够产生最大悬伸量的堆叠配置。

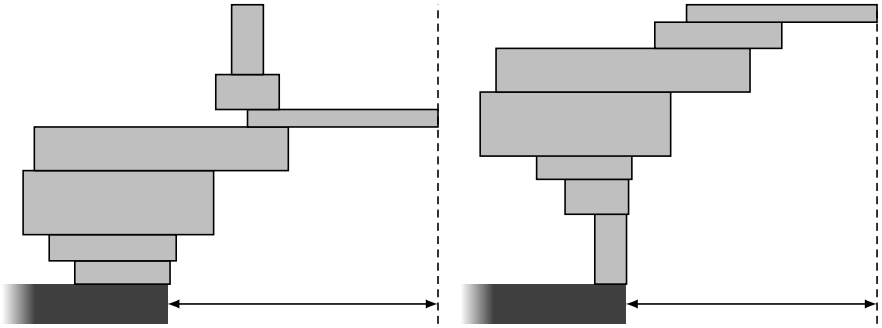


图 3：使用和平衡配重下的最优堆叠配置。在本图及后续所有图中，我们假设块的质量与其宽度和高度的乘积成正比。

在第3节中，我们将证明，由于任意宽度和质量带来的额外复杂性，积木堆叠问题实际上是 NP 难的。也就是说，除非 P $=$ NP，否则无法通过高效（即多项式时间）算法找到解，更不用说通过简单规则了。从某种程度上说，这是对 Treeby 针对高度相同的块提出的猜想的回应。

此外，这也说明，看似简单的问题（例如堆叠积木）可能出奇地棘手。不过，在第7节中，我们会给出一个（$2+\epsilon$）近似算法。 α 近似算法是指一种高效算法，它返回的解的目标值（在我们的情形中为悬伸量）与最优值的偏差至多为 α 倍。

不同于等高块的情形，即使不允许使用平衡配重，目前也没有已知的简单方法来确定可能的最大悬伸量。尽管如此，块堆叠问题、飞机加油问题和鲁棒预约调度问题这三个背景完全不同的优化问题之间，仍存在密切联系。

飞机加油问题是一个数学谜题，要求最大化一支能够在飞行中共享燃料的机队的航程。问题的详细描述见第 5 节。由于它与具有凹成本函数的单机调度问题有关（更多详情见 [25]），在 [26] 将其作为计算复杂性未知的问题提及后，该问题近年来引起了一些关注。飞机加油问题能否被高效求解仍是一个悬而未决的问题。然而，Gamzu 等人 [6] 最近开发了一种多项式时间近似方案（PTAS），对于任意固定的 $\epsilon > 0$，提供了一个$(1+\epsilon)$-近似算法。此外，研究人员还考察了支配条件和优先约束 [25]，并据此设计了一种分支定界算法 [15]。此外，还提出了一个带有 $O(n^2)$约束的混合整数线性规划模型 [27]。由于后两种方法在运行时间方面优于朴素方法，因此有助于解决实际问题。然而，从复杂性理论的意义上说，它们并不被认为是高效的（参见第3节）。尽管名为飞机加油问题，但迄今为止，它主要仍是一个学术谜题，没有直接的现实应用。

相比之下，预约调度是一类具有实际背景的优化问题，其目标是为时间需求只能估算的任务制定理想的日程安排（详情见第 6 节）。作为若干随机模型的替代方案，Mittal 等人 [17] 提出了鲁棒预约调度问题。正如 [17, 22], 所指出的，与飞机加油问题类似，鲁棒预约调度与具有凹成本函数的单机调度问题有关。因此，很快便得出了一些结果，涉及近似算法。简单的基于比率的算法 [9, 22],已经能够获得良好的近似保证；而且，与飞机加油问题一样，该问题也存在多项式时间近似方案（PTAS） [16, 23]。不过，对于鲁棒预约调度，能否高效地找到最优解仍然是一个开放问题。

尽管这些问题的起源各不相同，而且关于能否高效求解它们，人们所知甚少，但我们将在第5节和第6节证明，无平衡配重的积木堆叠、飞机加油和鲁棒预约调度在某种意义上是等价的，因此构成一组计算复杂性相同的问题。因此，飞机加油问题的现有算法也可用于求解鲁棒预约调度和无平衡配重的积木堆叠。此外，飞机加油问题的多项式时间近似方案（PTAS）

可以直接转用于积木堆叠问题。最重要的是，针对这三个问题中任意一个问题的高效算法，都意味着另外两个问题也存在这样的算法。

2 积木堆叠问题。

我们考虑一组 $B = \{1, ..., n\}$ 块，每个块由其半宽 $w_i \geq 0$ 和质量 $m_i > 0$刻画。目标是将这些块排列成一个平衡堆叠，使其尽可能远地伸出桌子的边缘之外。目前，设堆叠顺序固定，且这些块从上到下依次排列。对于每个块 i，令 x_i 表示其中点相对于桌子边缘的水平位置。此外，定义

$$M_i = \sum_{j:\, j \leq i} m_j$$

为顶部 i 个块的质量。根据物理定律，一个堆叠是平衡的，当且仅当对于每个 $i = 1$，.....， $n - 1$，块1至 i的重心 $\bar{x}_i = \left(\sum_{j:\, j \leq i} m_j x_j\right) / M_i$位于块 $i + 1$的边缘之内，即 $x_{i+1} - w_{i+1} \leq \bar{x}_i \leq x_{i+1} + w_{i+1}$。此外，不失一般性，我们假设整体重心恰好位于桌子的边缘。按照惯例，堆叠的悬伸量总是向右构建。

如相同木块的情形所示（参见图1），一种自然的策略是逐步将顶部的 i 块尽可能向右放置，同时避免堆叠倾倒。按照 Treeby [24], 的建议，我们称堆叠在块i 处右对齐，当且仅当第1块至 i 块的重心与块 $i + 1$ 的右边缘对齐，即 $\bar{x}_i = x_{i+1} + w_{i+1}$ （见图4）。然而，如前所述，不同积木的最优堆叠配置未必在每一块处都右对齐。因此，最大悬挑不一定由最上层积木达到，而是由 $\max_i(x_i + w_i)$ 给出。我们将右边缘达到这一悬伸量的块称为突出块。

我们希望找到悬伸量尽可能大的堆叠。通过推广 Treeby [24], 的命题5.1，可以明确描述任何此类堆叠必须具有的结构。在图1、图2和图3中已经可以观察到这种结构。

定理1。具有突出块 $p \in \{1, \ldots, n\}$的最大悬挑堆叠满足以下两个条件：

a) 堆叠中的所有块 $i \geq p$均为右对齐。

b) 块1至 $p - 1$ 的组合重心与突出块 p的左边缘对齐。

该证明紧密遵循了 [24], 中提出的思路，只需稍作调整即可将其应用于质量任意的块。

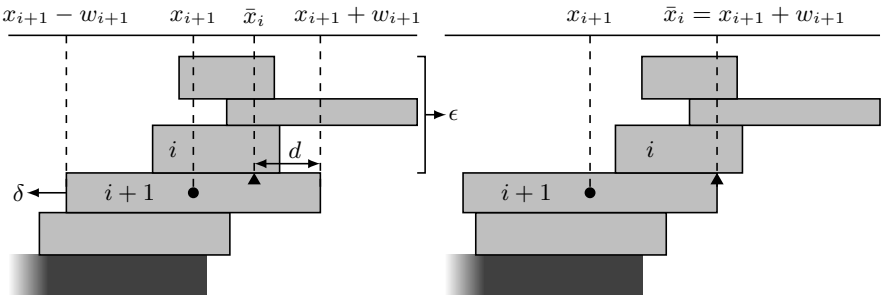


图4：两个叠堆都保持平衡，但只有右边的叠堆在块 i 处右对齐。按照定理1对左边叠堆所做的调整会带来更大的悬伸量。

证明。a) 假设堆叠在块 $i \geq p$ 处没有右对齐，如左侧的图4所示。则块1至 i 的重心位于块 $i+1$ 的右边缘左侧 $d > 0$ 的距离处。令 $\delta, \epsilon \in (0, d/2]$，使得 $\delta m_{i+1} = \epsilon M_i$。将块 $i+1$ 向左偏移 δ，并将块1至i（包括 p）向右移动 ϵ。根据 δ 和 ϵ 的定义，这一改动不会影响块1至 $i+1$ 的重心。因此，堆叠仍保持平衡，但悬伸量更大，这与原堆叠的最优性矛盾。

b) 假设块 1 到 $p-1$ 的重心位于 p 的左边缘右侧距离 $d > 0$ 处。令 δ, $\epsilon \in (0, d/2]$，使得 $\delta m_p = \epsilon M_{p-1}$。这一次，将块 1 到 $p-1$ 向左偏移 ϵ，并将 p 向右移动 δ。同样，叠堆仍保持平衡，且悬伸量比初始时更大。

□

根据定理1，如图5所示，任何最优堆叠配置都会将块分为配重块和右对齐块，其中最上方的右对齐块向外突出。配重块只是通过使突出块能够伸得更远，间接地增加悬伸量。另一方面，每个右对齐块 i 都会直接产生悬伸量，其大小等于组合重心 $\bar{x}_i$ 与该块右边缘 $x_i + w_i$ 之间的距离。简单计算可知，除突出块 p 外，该距离为 $w_i m_i / M_i$；由于配重块的作用，突出块对悬伸量的贡献距离为 $w_p(2 - m_p/M_p)$。因此，突出块 p 已知时，叠堆的最大悬伸量为

$$w_p\left(2 - \frac{m_p}{M_p}\right) + \sum_{i:\, i>p} \frac{w_i m_i}{M_i}, \quad \text{where} \quad M_i = \sum_{j:\, j \leq i} m_j. \tag{1}$$

请注意，配重越大，突出块对悬伸量的贡献越大；而对于其他右对齐块，压在其上的重量越大，效果越不利。这一点也可以从以下

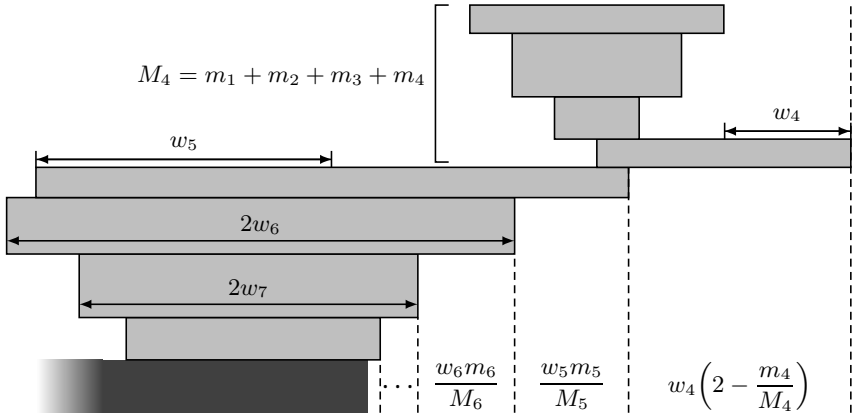


图5：最大悬挑堆叠的一般结构。顶部三个块用作配重，对悬伸量没有直接贡献。右对齐块逐步形成悬伸量。由于配重，突出块对悬伸量的贡献可以超过其宽度的一半。

示例。

2.1 堆叠配置。

考虑块 a 和 b，其参数为 $w_a = 1$、$m_a = k$、$w_b = k$ 和 $m_b = 1$，其中 $k \geq 2$。根据（1），若将块 a 放在顶部且不进行平衡配重，最大悬伸量为

$$w_a\left(2 - \frac{m_a}{m_a}\right) + w_b\frac{m_b}{m_a + m_b} = 1 + \frac{k}{k+1} = 2 - \frac{1}{k+1}.$$

另一方面，反转堆叠顺序，也就是将较长且较轻的块 b 放在顶部，则悬伸量可达到

$$w_b\left(2 - \frac{m_b}{m_b}\right) + w_a\frac{m_a}{m_b + m_a} = k + \frac{k}{1+k}.$$

请注意，对于第一种堆叠顺序，悬伸量不能超过 2，而后一种堆叠顺序产生的悬伸量超过其 $k/2$ 倍。因此，即使只考虑两个块且不使用平衡配重，重新排列堆叠顺序也能使结果提升任意大的倍数。

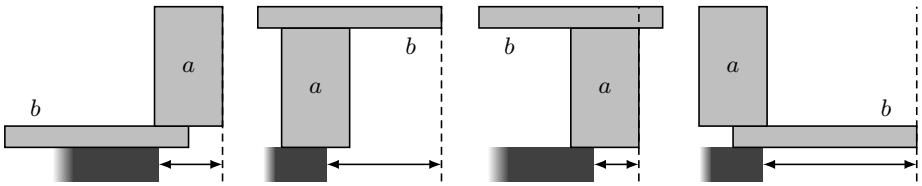


图 6：四种可能的最优堆叠配置，包括不使用和平衡配重的情况。

如果允许使用平衡配重，就可以采用另外两种堆叠配置（见图 6）。当配重为 m_b 时，块 a 的悬伸量为

$$w_a\left(2-\frac{m_a}{m_b+m_a}\right)=2-\frac{k}{k+1},$$

这甚至比第一个堆叠的悬伸量还小。不过，如果块 b 突出，而 a用作配重，则悬伸量为

$$w_b\left(2-\frac{m_b}{m_a+m_b}\right)=k\left(2-\frac{1}{k+1}\right)=2k-\frac{k}{k+1},$$

当 k 趋于无穷大时，悬伸量最多可达到不进行平衡配重时的两倍。

这个例子展示了重新排列块的妙处，也说明了寻找最优堆叠配置（无论是否采用平衡配重）的必要性。在接下来的章节中，我们将分析能否找到良好的堆叠配置，以及如何找到，并从几条简单的堆叠规则入手。

2.2 简单堆叠规则。

请注意，在图 6 的第三种堆叠中，一个又宽又轻的配重块伸得比原本应当突出的那个又短又重的块更远。当然，这样的堆叠不可能是最优的。事实上，从一开始就可以排除这种堆叠配置，因为绝不应使用又短又重的块作为突出块。本节将对这条规则及其他简单的堆叠规则加以形式化。

要理解以下论证，关键是弄清堆叠的悬伸量如何形成，以及例如交换两个块时悬伸量会如何变化。回顾式 (1)，右对齐块的贡献除取决于其自身参数外，只取决于其上方块的总质量。因此，交换两个块不会影响所有上方和下方块所产生的悬伸量。此外，请记住，计算悬伸量时，配重块只需考虑其总质量。

引理 2. 考虑一个最优堆叠配置，设 a、$b \in B$ 为两个右对齐块，且块 a 直接位于块 b 上方。那么

$$\frac{w_a}{M+m_a} \geq \frac{w_b}{M+m_b},$$

其中， M 是位于块 a上方的块的总质量。

证明。假设结论不成立。根据 (1)，交换块 a 和 b 会导致悬伸量发生如下变化:

$$\frac{w_b m_b}{M+m_b} + \frac{w_a m_a}{M+m_b+m_a} - \left(\frac{w_a m_a}{M+m_a} + \frac{w_b m_b}{M+m_a+m_b}\right).$$

将所有四个分数通分，并对所得项进行适当分组后，可得其等于

$$\begin{aligned}&\frac{w_b m_b m_a}{(M+m_b)(M+m_a+m_b)} - \frac{w_a m_a m_b}{(M+m_a)(M+m_b+m_a)}\\ &= \frac{m_b m_a}{M+m_b+m_a} \cdot \left(\frac{w_b}{M+m_b} - \frac{w_a}{M+m_a}\right) > 0.\end{aligned}$$

这与原堆叠配置的最优性矛盾。 □

备注。请注意，此条件不足以确定右对齐块的最佳堆叠顺序，以下反例对此进行了说明。设 $w_1 = 11$、 $w_2 = 21$、 $w_3 = 33$和 $m_1 = 1$、 $m_2 = 2$、 $m_3 = 4$ 分别为完全右对齐堆叠中块 1、2、3 的参数。那么，堆叠顺序 1、2、3 满足该条件，但如图 7 所示，序列 2、3、1 给出了严格更优的解。

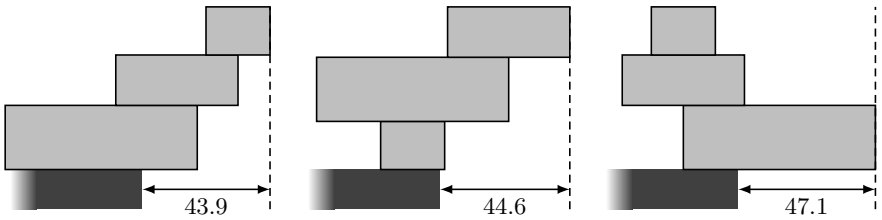


图 7： (a) 和 (b) 均满足引理 2 的条件，但 (b) 的悬伸量稍大。若允许使用平衡配重，则堆叠配置 (c) 为最优。

推论 3. 设 a、$b \in B$ 是最优堆叠配置中的两个右对齐块。若 $w_a > w_b$ 且 $w_a/m_a > w_b/m_b$，则块 a 不能直接放置在 b 下方。特别是，若 $w_a > w_b$ 且 $m_a \leq m_b$，则满足这一条件。

证明。

$$\frac{w_a}{M+m_a}-\frac{w_b}{M+m_b}=\frac{M(w_a-w_b)+(w_am_b-w_bm_a)}{(M+m_a)(M+m_b)}>0$$

对于任意 $M\geq 0$。该结论由引理 2 得出。 □

将推论 3 推广到 $w_a/m_a=w_b/m_b$ 的情形后，很容易看出，质量与宽度成正比的右对齐块，总能按宽度排序，最宽的放在最上方，从而形成最大悬挑的堆叠。如前所述，这一点由 Treeby [24] 证明过，并且可以从图 2 中的两个叠堆看出。

更一般地说，推论 3 确认，对于右对齐块，较宽且较轻的块应放在较短且较重的块上方。特别是，如果某个块比所有其他块都宽，且质量至少不比它们重，那么推论 3 意味着该块不能放在任何右对齐块下方。在这种情况下，我们甚至可以证明，它也不能用作配重。因此，为了使堆叠达到尽可能大的悬伸量，该块必须伸出。

引理 4。设 $b_*\in B$ 是一个块，对于所有 $b\in B\setminus\{b_*\}$，$b_*\in B$ 具有 $w_{b_*}>w_b$ 和 $m_{b_*}\leq m_b$。那么，在任何最优堆叠配置中，b_* 都会突出。

证明。考虑一个 b_* 不突出的堆叠配置。根据推论 3，b_* 不可能是一个不突出的右对齐块。因此，b_* 必须在突出块 $p\in B\setminus\{b_*\}$ 的上方充当配重块。设 M 为其他配重块的总质量。现在考虑交换 b_* 和 p 且使 b_* 突出的堆叠配置。请注意，这一变化不会影响突出块下方的块所产生的悬伸量（参见（1））。由于 $m_{b*}\leq m_p$，新的堆叠配置所实现的悬伸量大于原配置：

$$\begin{aligned}&w_{b_*}\left(2-\frac{m_{b_*}}{M+m_p+m_{b_*}}\right)-w_p\left(2-\frac{m_p}{M+m_{b_*}+m_p}\right)\\&\geq\underbrace{(w_{b_*}-w_p)}_{>0}\cdot\underbrace{\left(2-\frac{m_p}{M+m_{b_*}+m_p}\right)}_{\geq 1}>0.\end{aligned}$$

因此，在最优堆叠配置中，b_* 既不是配重块，也不是不突出的右对齐块。 □

还可以获得与此结果类似的其他见解。然而，在下一节中，我们将证明，尽管对积木堆叠问题已有这些认识，要找到最优堆叠配置仍然是一个可证明的难题。

3 NP-困难性。

对于固定的堆叠顺序和指定的突出块 p，可实现的悬伸量可由公式 (1) 得到。积木堆叠问题的目标是找出一种能实现最大悬伸量的堆叠配置。根据 (1)，利用双射函数 $\pi\colon B \to \{1,\dots,|B|\}$ 来表示堆叠顺序（从上到下），可将问题简洁地表述为

$$\max_{p\in B} \max_{\pi} \; w_p\left(2-\frac{m_p}{M_p}\right)+\sum_{i\colon \pi(i)>\pi(p)} \frac{w_i m_i}{M_i}, \quad \text{where} \quad M_i=\sum_{j\colon \pi(j)\le\pi(i)} m_j. \tag{2}$$

由于有 $n\cdot n!$ 种可能的堆叠配置，也就是说，可能的堆叠配置数量呈指数级增长，因此计算所有配置的潜在悬伸量极为低效，即使 n 相当小，也会超出任何现有计算能力。上一节讨论的结果或许能在一定程度上缩短求得最优解所需的时间，但这些改进仍不足以设计出高效的算法。

在复杂性理论中，如果对于任何可能的输入，所需的计算步骤数都是输入参数规模的多项式函数，就认为该算法是高效的。满足这一条件的算法也称为多项式时间算法。许多有趣的问题至今都没有找到多项式时间算法，例如著名的旅行商问题。这促成了所谓 NP 困难问题这一复杂性类的建立。如果其中任何一个问题能在多项式时间内求解，那么所有这些问题都能求解（此时 P = NP，而这被认为不太可能）。

本节将证明积木堆叠是 NP 困难的。为此，我们将说明，任何高效求解积木堆叠问题的算法，都可用于高效解决一个已知的 NP 困难问题。我们将证明，若能在多项式时间内求解积木堆叠问题，就能在多项式时间内求解 NP 困难的划分问题。关于复杂性理论的详细介绍以及划分问题的 NP-困难性证明，请参见例如 [5]。

定义（划分问题）。给定一组正整数 $A=\{a_1,\dots,a_n\}$，判断能否将这些元素划分为两个不相交子集 $A_1 \cup A_2 = A$，使两个子集中的元素之和相等，即 $\sum_{a_i\in A_1} a_i = \sum_{a_i\in A_2} a_i$。

若存在这样的划分，则称其为完美划分。不失一般性，我们可以假设 $\sum_{a_i\in A} a_i$ 为偶数。因此，令 $T=\frac{1}{2}\sum_{a_i\in A} a_i \ge 1$表示目标分割。

3.1 通过积木堆叠问题求解划分问题。

给定划分问题的一个实例 $A=\{a_1,\dots,a_n\}$，考虑以下一组块 B：

- 对于 $i = 1, \ldots, n$，设 $i \in B$ 为一个质量为 $m_i = a_i$、半宽为$w_i = 1$的块，
- 设 $b^\bullet \in B$ 为一个质量为 $m_\bullet = 1$、半宽为 $w_\bullet = \left(2T + \frac{5}{4}\right)^5$的块，并且
- 设 $b^* \in B$ 为一个质量为 $m_* = 1/4$、半宽为

$$w_* = \frac{w_\bullet m_\bullet}{m_*} \frac{(T + m_*)^2}{(T + m_* + m_\bullet)^2} = 4w_\bullet \left(1 - \frac{1}{T + \frac{5}{4}}\right)^2.$$

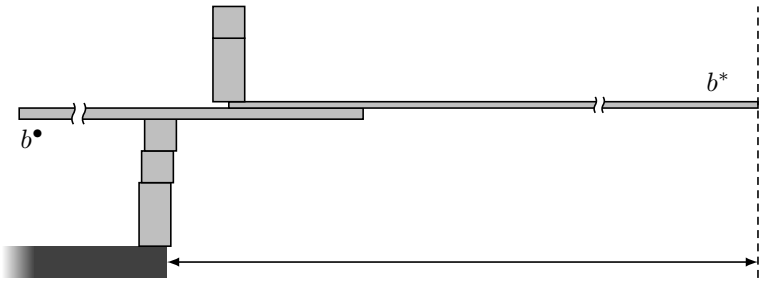


图 8: $\bullet*$-悬挑堆叠的示意结构。 $b^\bullet$ 和 b^* 共同将其余块划分为配重块和右对齐块。

我们的目标是证明，对这组块求解积木堆叠问题，就能自动得出相应划分问题实例的答案。更具体地说，我们要证明，在任何最优堆叠配置中，块 b^* 都会悬挑，而 $b^\bullet$ 则直接放在其下方。因此，如图 8 所示，它们共同将块 $1, \ldots, n$ 划分为配重块和右对齐块。我们将这种结构的堆叠称为 $\bullet*$- 悬挑堆叠。块的重量和质量经过精心选择，既满足引理 5 和 7 的所有条件，也尽可能简化了相应的证明。虽然使用两个辅助块为这些参数的选择提供了一定自由度，但请注意，使用单个辅助块无法实现这种构造。在说明这有助于找到完美划分之前，我们先证明，给定积木堆叠问题实例的任何最优解确实都是 $\bullet*$-悬挑的。

引理 5。任何最优堆叠配置都是 $\bullet*$-突出式的。

证明。我们将分三步证明这一结论。在任何最优堆叠配置中

a) b^*突出，b) $b^\bullet$ 是一个右对齐块，
且 c) $b^\bullet$ 直接放置在 b^*下方。

a) 根据引理 4，只需证明 $w_* > w_\bullet$、 $w_* > w_i$和 $m_* \le m_\bullet$、 $m_* \le m_i$ 对所有 $i \in B \setminus \{b^*, b^\bullet\}$都成立。后者显然成立，因为 $m_* = 1/4 < 1 = m_\bullet \le m_i$。此外，

$$w_* = 4w_\bullet\Big(1 - \frac{1}{T + \frac{5}{4}}\Big)^2 \ge 4w_\bullet\Big(1 - \frac{1}{1 + \frac{5}{4}}\Big)^2 > w_\bullet = \big(2T + \tfrac{5}{4}\big)^5 > 1 = w_i.$$

b) 假设 $b^\bullet$ 充当配重。首先，考虑只有 $b^\bullet$ 是配重块的情况。这意味着， b^* 正下方有一个右对齐块 $i \in B \setminus \{b^*, b^\bullet\}$。交换 $b^\bullet$ 和 i 会使悬伸量增加

$$w_*\Big(\underbrace{\frac{m_i}{m_i + m_*} - \frac{m_\bullet}{m_\bullet + m_*}}_{\ge 0 \text{ as } m_i \ge m_\bullet}\Big) + \frac{w_\bullet m_\bullet - w_i m_i}{m_* + m_\bullet + m_i} \ge \frac{\big(2T + \frac{5}{4}\big)^5 - 2T}{m_* + m_\bullet + m_i} > 0.$$

现在考虑配重块（不包括 $b^\bullet$）的总质量为 $M \ge 1$的情况。将 $b^\bullet$ 移到 b^* 正下方会使悬伸量增加

$$\begin{aligned}
&w_*\Big(2 - \frac{m_*}{M + m_*}\Big) + \frac{w_\bullet m_\bullet}{M + m_* + m_\bullet} - w_*\Big(2 - \frac{m_*}{M + m_* + m_\bullet}\Big) \\
&= \frac{w_\bullet m_\bullet}{M + m_* + m_\bullet} + w_*\Big(\frac{m_*}{M + m_* + m_\bullet} - \frac{m_*}{M + m_*}\Big) \\
&= \frac{m_\bullet}{M + m_* + m_\bullet} \cdot \Big(w_\bullet - \frac{w_* m_*}{M + m_*}\Big),
\end{aligned}$$

该值严格大于零，因为 $M \ge 1$ 和

$$w_* m_* = w_\bullet\Big(1 - \frac{m_\bullet}{T + m_* + m_\bullet}\Big)^2 < w_\bullet.$$

c) 由于对于所有 $i \in B \setminus \{b^*, b^\bullet\}$， $w_\bullet > w_i$ 和 $m_\bullet \le m_i$，由推论 3 可立即得出，在最优解中， $b^\bullet$ 不能放置在这些块中的任何一个下方。既然我们已经看到 $b^\bullet$ 是一个右对齐块，它就必须紧贴在突出块 b^*的正下方。因此，任何最优堆叠配置都是 $\bullet*$-突出型。 □

根据引理 5，对于任何具有最大悬挑的堆叠配置， $b^\bullet$ 和 b^*将块 $1, \dots, n$ 划分为配重块和右对齐块。给定任意一组数字 $a_1, \dots, a_n$，现在我们要证明：如果存在完美划分，那么将对应的块 $1, \dots, n$ 划分为配重块和右对齐块各自的总质量均为 T时，效果最佳。为此，我们假设元素$2T = \sum_{a_i \in A} a_i$ 的总和固定，并引入函数 $O_{\min}(C)$和 $O_{\max}(C)$，分别表示给定配重质量 $C \in \{0, \dots, 2T\}$时可实现的最小和最大悬伸量。为了求出 $O_{\min}(C)$和 $O_{\max}(C)$的显式表达式，我们证明以下引理。

引理 6. 将块 $i \in B \setminus \{b^*, b^\bullet\}$ 替换为两个半宽相同的块 i_1、 i_2，其质量为 $m_{i_1} + m_{i_2} = m_i$，则悬伸量不会减小。

证明. 如果 i 用作配重，悬伸量保持不变。否则，设 M 为 i上方所有块的质量。将 i 替换为 i_1 和 i_2，其中 i_1 位于 i_2上方，则堆叠的悬伸量不会减小：

$$\frac{w_{i_1} m_{i_1}}{M + \underbrace{m_{i_1}}_{\leq m_i}} + \frac{w_{i_2} m_{i_2}}{M + \underbrace{m_{i_1} + m_{i_2}}_{= m_i}} - \frac{w_i m_i}{M + m_i} \geq w_i \, \frac{m_{i_1} + m_{i_2} - m_i}{M + m_i} = 0.$$

□

根据引理 6，当右侧对齐的块尽可能多地拆分时， $\bullet*$-悬挑堆叠的最大悬伸量最大。由于 a_1、.....、a_n只能是正整数，相应堆叠的悬伸量至多等于这样一种 $\bullet*$-悬挑堆叠的悬伸量，其中所有常规块的质量均为$m_i = 1$。因此，对于配重 C,

$$O_{\max}(C) = w_* \left(2 - \frac{m_*}{C + m_*}\right) + \frac{w_\bullet m_\bullet}{C + m_* + m_\bullet} + H_C,$$
$$\text{where} \quad H_C = \sum_{i=1}^{2T-C} \frac{1}{C + m_* + m_\bullet + i} \ .$$

反过来，如图 9 所示，当 $\bullet*$-悬挑结构下方只有一个质量为 $2T - C$ 的块时，悬伸量最小。因此,

$$O_{\min}(C) = w_* \left(2 - \frac{m_*}{C + m_*}\right) + \frac{w_\bullet m_\bullet}{C + m_* + m_\bullet} + \frac{2T - C}{2T + m_* + m_\bullet} \ .$$

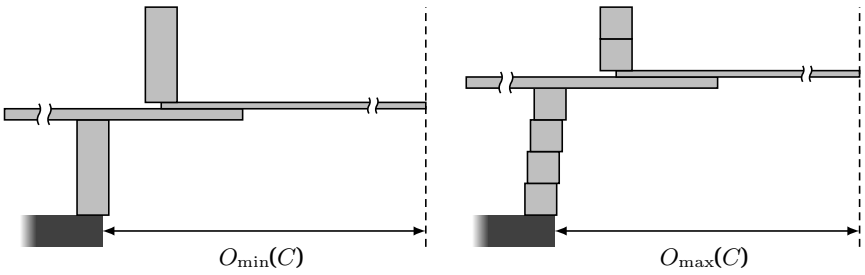


图 9：具有最小和最大悬伸量的 $\bullet*$-悬挑堆叠结构。根据引理 7，左侧堆叠的悬伸量大于任何不存在完美划分的堆叠，例如右侧所示的堆叠。

为证明最大悬挑的堆叠配置依赖于完美划分，我们证明，即使采用配重 T，最差的堆叠配置仍比任何采用不同划分的堆叠配置具有更大的悬伸量。

引理 7. 对所有 $K = 1, \ldots, T$， $O_{\min}(T) > O_{\max}(T \pm K)$。

证明. 只需证明

$$w_*\left(2-\frac{m_*}{T+m_*}\right)+\frac{w_\bullet m_\bullet}{T+m_*+m_\bullet}$$
$$>w_*\left(2-\frac{m_*}{T\pm K+m_*}\right)+\frac{w_\bullet m_\bullet}{T\pm K+m_*+m_\bullet}+H_{T\pm K},$$

或者，等价地,

$$\mp\frac{Kw_*m_*}{(T+m_*)(T\pm K+m_*)}\pm\frac{Kw_\bullet m_\bullet}{(T+m_*+m_\bullet)(T\pm K+m_*+m_\bullet)}>H_{T\pm K}\ .$$

根据 w_* 的定义,

$$\begin{aligned}&\mp\frac{Kw_*m_*}{(T+m_*)(T\pm K+m_*)}\pm\frac{Kw_\bullet m_\bullet}{(T+m_*+m_\bullet)(T\pm K+m_*+m_\bullet)}\\&=\mp\frac{Kw_\bullet m_\bullet}{(T+m_*+m_\bullet)^2}\left(\frac{T+m_*}{T\pm K+m_*}-\frac{T+m_*+m_\bullet}{T\pm K+m_*+m_\bullet}\right)\\&=\frac{K^2w_\bullet m_\bullet^2}{(T+m_*+m_\bullet)^2(T\pm K+m_*+m_\bullet)(T\pm K+m_*)}\ .\end{aligned}$$

由于 $1\le K\le T$、 $m_\bullet=1$ 和 $w_\bullet=(2T+m_*+m_\bullet)^5,$, 这大于 $2T$，而后者又大于 $H_{T\pm K}$。 □

定理 8. 求解带平衡配重的积木堆叠问题的最优堆叠配置是 NP 困难的。

证明. 给定一组用于划分问题的正整数 $A=\{a_1,\dots,a_n\}$，考虑上述定义并分析过的积木堆叠问题实例。注意，此实例及其所有参数的规模均为多项式规模。根据引理 5 和 7，显然， A 存在完美划分，当且仅当积木堆叠问题的所有最优解都向 $\bullet*$ 方向突出，并且使用恰好为 T 的配重。给定一个能够最优求解积木堆叠问题的堆叠配置，可以高效地计算其配重。因此，积木堆叠是 NP 困难的。

□

4 无平衡配重的积木堆叠。

在上一节中，我们发现一般而言，积木堆叠是 NP 困难的，因此要高效地确定最优堆叠配置实际上难以实现。然而，如果只允许不使用平衡配重的堆叠，也就是像右侧图 3 所示的完全右对齐堆叠，情况又会如何呢？对于质量与宽度成正比的块，Treeby [24] 证明，不使用平衡配重时，积木堆叠问题的分析要简单得多，只需按块的宽度排序即可求解（参见[24], 推论 3 中的引理 4.2 和图 2）。对于质量任意的块，似乎不存在这样简单的算法

，而能否高效地找到精确解仍是一个悬而未决的问题。尽管如此，这个变体仍有许多值得探讨之处。归根结底，接下来的讨论甚至将有助于为一般积木堆叠问题（即允许使用平衡配重的情况）找到一种近似算法。

当堆叠完全右对齐时，突出块总是位于最上方的块。因此，只需优化堆叠顺序，且（2）可简化为

$$\max_{\pi} \sum_{i \in B} \frac{w_i m_i}{M_i}, \quad \text{where} \quad M_i = \sum_{j:\, \pi(j) \le \pi(i)} m_j. \tag{3}$$

此前尚未研究过这一版本的积木堆叠问题。事实证明，它在结构上与两个背景截然不同的问题相似。接下来的两节将证明积木堆叠、飞机加油和鲁棒预约调度之间存在密切联系。

5 飞机加油问题。

飞机加油问题也称为吉普车队问题、吉普车纵队问题或 n 车辆探索问题，其起源是以下数学谜题（参见 [3, 25, 26]）。假设有一些货物需要运送到尽可能远的地方，超出单架飞机所能抵达的距离。因此，需要派遣一支机队来完成任务。飞机之间可以在飞行中即时交换燃料。当一架飞机燃料耗尽，或已将所有燃料转交给其他飞机时，它便退出，不再计入燃料消耗。因此，给定每架飞机的燃油消耗率和油箱容量，飞机加油问题旨在确定一个退出序列，使最后一架留下的飞机能够飞行尽可能远的距离。或者说，吉普车队问题 [20] 和 n车辆探索问题 [27] 所考虑的是使用吉普车车队穿越沙漠的任务。然而，从数学上看，它们与飞机加油问题完全相同。需要注意的是，吉普车问题也指一些仅考虑单辆车的变体（例如，参见 [2, 4, 8, 13, 20]）。为清晰起见，本文余下部分将沿用飞机加油这一背景。

形式化地，设 $A = \{1, \ldots, n\}$ 是一组飞机，每架飞机由其油箱容量 $v_i \ge 0$ 和燃油消耗率 $c_i > 0$ 指定。当然，不应浪费任何燃料；除此之外，为了降低总燃料消耗，应尽可能早地淘汰每架飞机。相应地，当继续飞行的飞机能够独自携带所有剩余燃料时，最优做法是立即让一架飞机停止飞行。按照这一流程，机队在退出

序列 σ 下为: $\{1,\dots,n\}\to A$, 一系列

$$\frac{v_{\sigma(1)}}{c_{\sigma(1)}+\cdots+c_{\sigma(n)}}+\cdots+\frac{v_{\sigma(n-1)}}{c_{\sigma(n-1)}+c_{\sigma(n)}}+\frac{v_{\sigma(n)}}{c_{\sigma(n)}}=\sum_{i=1}^{n}\frac{v_{\sigma(i)}}{\sum_{j=i}^{n}c_{\sigma(j)}}\ .$$

定义 π: $A\to\{1,\dots,n\}$ 为 σ 的逆, 并注意到

$$\sum_{i=1}^{n}\frac{v_{\sigma(i)}}{\sum_{j=i}^{n}c_{\sigma(j)}}=\sum_{i\in A}\frac{v_i}{C_i},\quad\text{where}\quad C_i=\sum_{j:\,\sigma^{-1}(j)\geq\sigma^{-1}(i)}c_j=\sum_{j:\,\pi(j)\geq\pi(i)}c_j.$$

因此, 要使机队航程最大化, 归结为求解

$$\max_{\pi}\sum_{i\in A}\frac{v_i}{C_i},\quad\text{where}\quad C_i=\sum_{j:\,\pi(j)\geq\pi(i)}c_j. \tag{4}$$

或者, 将 π': $A\to\{1,\dots,n\}$ 定义为 π的反向, 即对于所有 $i\in A$, $\pi'(i)=n+1-\pi(i)$, 则飞机加油问题为

$$\max_{\pi'}\sum_{i\in A}\frac{v_i}{C_i},\quad\text{where}\quad C_i=\sum_{j:\,\pi'(j)\leq\pi'(i)}c_j. \tag{5}$$

将最后一种表述与式（3）中的无平衡配重的积木堆叠进行比较, 可以发现二者有一个有趣的相似之处。事实上, 对于所有 $i\in A$, 令$A=B$、$c_i=m_i$ 和 $v_i=w_im_i$ 对应, 则求解（5）等价于求解（3）。反过来, 将对于所有 $i\in B$的 $B=A$、 $m_i=c_i$和 $w_i=v_i/c_i$ 设定为相应值, 飞机加油问题便可建模为积木堆叠问题的一个实例。因此, 积木堆叠问题的任一最优堆叠顺序, 都对应于相应飞机加油问题实例的一个最优退出序列, 反之亦然。

定理 9. 不含平衡配重的积木堆叠问题等价于飞机加油问题。

因此, 这两个问题具有相同的计算复杂性。此外, 针对其中一个问题已知的任何算法（包括近似算法）都可以立即应用于另一个问题。特别地, 回想一下, 飞机加油问题允许使用多项式时间近似方案 [6]。也就是说, 对于每个 $\epsilon>0$, 都存在一个 $(1+\epsilon)$-近似算法。

推论 10. 存在一种用于不含平衡配重的积木堆叠问题的多项式时间近似方案。

6 鲁棒预约调度。

现在我们来研究鲁棒预约调度, 这是一类看似无关的应用导向型优化问题。尽管如此, 我们将建立它与飞机加油问题之间的联系。因此, 鲁棒预约调度和

飞机加油和积木堆叠构成了一组联系紧密、十分显著的问题。

在服务业中，尤其是在医疗保健等成本高昂的领域，尽可能高效地安排预约调度可以带来显著优势。然而，由于医疗任务所需的时间通常只能估算，任何预先制定的日程安排都不可避免地会导致部分任务早于或晚于分配的时间完成。尽管无法避免，这两种情况都不理想。如果任务提前完成，人员、设备和设施就会闲置，直到下一项任务开始，由此产生利用不足成本，因为这段时间原本可以分配给其他任务。另一方面，任务延迟完成不仅会因自身的延误产生额外成本（我们将其称为该任务的超额成本），还会导致下一项任务无法按预定时间开始。因此，过于紧凑的日程安排可能因延误不断累积而造成高昂成本。预约调度问题旨在找到一种日程安排，使可能产生的利用不足成本与延误成本达到最优平衡。请注意，降低成本的关键一环是调整任务的处理顺序。由于日程安排的不确定性会随着此前任务数量的增加而上升，因此，应优先处理相关成本较高的任务。同时，将不确定性较低的任务安排在前面，可以提高整个日程安排的可靠性，从而降低总体成本。当现有任务无法重新排序以同时满足这两项原则时，问题便会变得复杂。

下面我们考虑 Mittal 等人提出的模型，该模型基于鲁棒优化。对于一组作业（即任务） $J = \{1, \ldots, n\}$，每个作业都由可能处理时间的区间 $[\underline{p}_i, \overline{p}_i]$、单位超额成本 $o_i > 0$和共同的单位未充分利用成本 $u > 0$表征，目标是找到一个日程安排，包括处理顺序 $\pi\colon\ J \to \{1, \ldots, n\}$ 以及分配给每项作业 $i \in J$ 的时间段 t_i ，使最坏情形下的成本最小。该模型的扩展和变体已在 [1, 22]中探讨。

正如 [17], 所证明的，对于固定的处理顺序 π，t_1，.....，t_n 的最优值是相应作业处理时间上下界的加权平均值。对于$\Delta_i = \overline{p}_i - \underline{p}_i$，它们为

$$t_i = \frac{u\underline{p}_i + O_i\overline{p}_i}{u + O_i} = \underline{p}_i + \Delta_i \frac{O_i}{u + O_i}, \quad \text{where} \quad O_i = \sum_{j\colon \pi(j) \geq \pi(i)} o_j.$$

此外，最坏情形的总成本也可以用闭式表示。因此，寻找最佳处理顺序就变成如下优化问题:

$$\min_{\pi} \sum_{i \in J} \Delta_i \frac{uO_i}{u + O_i}, \quad \text{where} \quad O_i = \sum_{j\colon \pi(j) \geq \pi(i)} o_j.$$

注意，无论处理顺序 π如何，都有

$$\sum_{i\in J}\Delta_i\frac{uO_i}{u+O_i}+u^2\sum_{i\in J}\frac{\Delta_i}{u+O_i}=\sum_{i\in J}\Delta_i\frac{uO_i+u^2}{u+O_i}=u\sum_{i\in J}\Delta_i.$$

由于右侧与 π 无关，因此最小化左侧第一项求和等价于最大化第二项求和。此外，由于 $u>0$，我们有：

引理 11. 当且仅当处理顺序 π 对鲁棒预约调度问题最优时，它也是

$$\max_{\pi}\sum_{i\in J}\frac{\Delta_i}{u+O_i},\quad where\quad O_i=\sum_{j:\,\pi(j)\geq\pi(i)}o_j. \tag{6}$$

将式 (6) 与式 (4) 中给出的飞机加油问题进行比较可知，若非式 (6) 分母中的偏移量 $u>0$，这两个问题本可等价。我们仍利用这种相似性来证明，可以用飞机加油问题求解式 (6)。

这一变换的思路是，将不确定性范围为 Δ_i、超额成本为 o_i 的作业对应到油箱容量为 $v_i=\Delta_i$ 、燃油消耗率为 $c_i=o_i$ 的飞机。接下来，只需通过添加一架燃油消耗率为 $c_*=u$ 的辅助飞机 a^*，将偏移量 $u>0$ 纳入累计燃油消耗率 C_i 。为确保 u 加到所有 C_i 上，必须使辅助飞机在任何最优解中最后被移除；将其油箱容量 v_* 设得足够大即可保证这一点。为更正式地说明这一点，我们需要飞机加油问题的以下必要最优性条件。利用第 5 节中建立的与积木堆叠问题的联系，不难看出，这实质上是对引理 2 的重新表述。

引理 12. 假设飞机 $A=\{1,\dots,n\}$ 按最优脱落顺序编为索引，即对于所有 $i\in A$，均有 $\pi(i)=i$ 。那么，对于累计燃油消耗率 C_{n+1}：$=0<C_n<\cdots<C_2<C_1$，对于所有 $i=2,\dots,n$，均成立

$$f_i(C_{i+1})\geq f_{i-1}(C_{i+1}),\quad where\quad f_i(x)=\frac{v_i}{c_i(x+c_i)}\ .$$

引理 13. 设 v_i, c_i 分别为飞机 $i\in A$ 的油箱容量和燃油消耗率，其中至少有一个 $i\in A$ 满足 $v_i\neq 0$ 。给定一架燃油消耗率为 $c_*>0$ 的额外飞机a^*，则存在一个多项式规模的油箱容量 v_*，使得在任何最优脱落顺序中，额外飞机都最后退出。特别地，

$$v_*=\frac{c_*\max_j v_j}{c_\bullet^2}\left(c_*+\sum_{j\in A}c_j\right),$$

其中 $c_\bullet=\min\{c_*,c_j\mid j\in A\}$ 满足这些要求。

证明。根据 v_* 和 $c_\bullet$ 的定义，对所有 $i \in A$ 和 $x \in [0, \sum_{j\in A} c_j]$，均有

$$f_*(x) = \frac{v_*}{c_*(c_*+x)} = \frac{\max_j v_j}{c_\bullet^2} \underbrace{\frac{c_*(c_*+\sum_{j\in A} c_j)}{c_*(c_*+x)}}_{\geq 1} \overset{\text{(I)}}{\geq} \frac{v_i}{c_i^2} \overset{\text{(II)}}{\geq} \frac{v_i}{c_i(c_i+x)} = f_i(x).$$

事实上，我们甚至可以看出 $f_*(x) > f_i(x)$，因为 (I) 仅当$v_i = \max_j v_j$且 $x = \sum_j c_j$时才能取等号，而 (II) 当且仅当 $x = 0$或 $v_i = 0 < \max_j v_j$时取等号。由于 $C_{a^*} - c_*$必须位于 $[0, \sum_{j\in A} c_j]$内，引理 12 表明，在任何最优解中，额外的飞机都必须最后退出。 □

引理 14. 若存在一个求解飞机加油问题的多项式时间算法，则也存在一个求解 (6) 的多项式时间算法。

证明。考虑一个由 $\Delta_i \geq 0$, $o_i > 0$ （对所有作业 $i \in J$）以及偏移量 $u > 0$给出的 (6) 的实例。不失一般性，我们可以假设至少对于某个 $i \in J$，有 $\Delta_i \neq 0$；否则，求解 (6) 是显然的。令 $A = J \cup \{a^*\}$ 为一组飞机，其规格如下。对于每个 $i \in J$，考虑一架油箱容量为 $v_i = \Delta_i$ 、燃油消耗率为 $c_i = o_i$的飞机。此外，按照引理 13 中的规定，加入一架燃油消耗率为 $c_* = u$ 、油箱容量为v_*的辅助飞机 a^* 。因此，在该飞机加油问题实例的任何最优解中，这架飞机都必须最后退出。因此，任意最优退出顺序 $\pi^*\colon A \to \{1, \ldots, n+1\}$ 均满足 $\pi^*(a^*) = n+1$ 且$\pi^*(i) = \pi(i)$，对所有 $i \in J$以及某个排列 $\pi\colon J \to \{1, \ldots, n\}$成立。令 C_i为飞机 $i \in A$的累计燃油消耗率，令 O_i 为相对于 π的作业 $i \in J$ 的累计超额成本。则对所有 $i \in J$， $O_i = C_i - c_*$，并且

$$\sum_{i\in A} \frac{v_i}{C_i} = \frac{v_*}{C_{a^*}} + \sum_{i\in J} \frac{v_i}{c_* + (C_i - c_*)} = \frac{v_*}{c_*} + \sum_{i\in J} \frac{\Delta_i}{u + O_i}.$$

由于 v_*/c_* 是常数，对于给定的飞机加油问题实例，排列 π 是最优的，

$$\max_\pi \sum_{i\in A} \frac{v_i}{C_i},$$

当且仅当它对于 (6) 是最优的，也就是说，对于

$$\max_\pi \sum_{i\in J} \frac{\Delta_i}{u + O_i}.$$

根据引理 13， v_* 相对于输入参数而言具有多项式规模。因此，通过多项式时间算法求解给定的飞机加油问题实例，就构成了求解 (6) 的多项式时间算法。 □

也可以证明逆向结论。在引理 14 的证明中，偏移量 u被建模为一架额外的飞机。反过来，我们可以移除一架飞机

，一次移除一架，并利用偏移量模拟其燃油消耗率。求解 (6) 即可得到飞机加油问题的最优排列，前提是被移除的飞机必须最后退出。通过让每架飞机依次处于这种情形，并比较由此得到的目标值，即可确定整体最优退出顺序。这个证明思路很容易形式化，但由于它与引理 14 的证明相似，这里略去。不过，结合引理 11，我们可以得出以下结论。

定理 15. 鲁棒预约调度存在多项式时间算法，当且仅当飞机加油问题存在多项式时间算法。

这两个问题之间的联系，使我们能够利用飞机加油问题的现有算法来求解鲁棒预约调度。然而，由于目标函数存在偏移量，近似算法无法直接迁移。尽管如此，如前所述，鲁棒预约调度已经有了优秀的近似算法。根据定理 9，鲁棒预约调度不仅与飞机加油问题有关，也与积木堆叠有关。由此，这些问题构成了一组具有相同计算复杂性的难题，尽管它们的真实复杂性仍未可知。这表明，没有明显应用价值的数学谜题（例如飞机加油问题和积木堆叠）取得的进展，有时也能直接影响预约调度等现实问题。

7 积木堆叠的 $(2+\epsilon)$-近似算法。

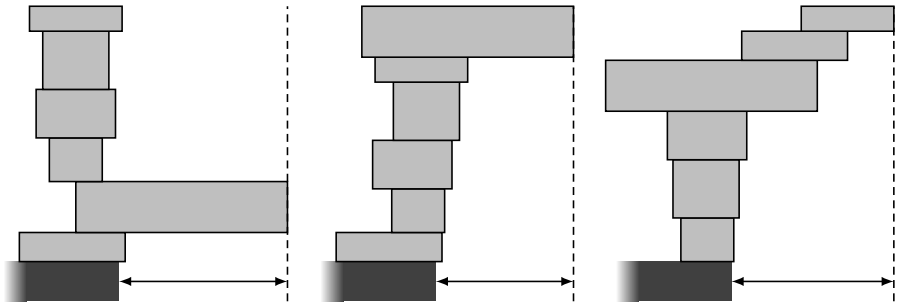
在第3节中，我们看到带平衡配重的积木堆叠问题是 NP 困难的。因此，要找到最优解的多项式时间算法并非易事。尽管如此，我们仍可通过近似算法高效地求得有保证的良好解。在第1节和第2节中，我们注意到，配重块有助于增大悬伸量。然而，这种差异通常很小。此外，无论是否使用平衡配重，我们总是希望让宽而轻的块突出。因此，在许多情况下，两种情形的最优堆叠配置中，突出的都是同一块，例如图 2 和图 3 所示。虽然并非对所有块集合都如此（见图 7），但我们可以证明，对于任意堆叠配置，都存在一个突出块相同的完全右对齐堆叠，其悬伸量至少是原配置的一半。具体而言，这意味着，求出最优的完全右对齐堆叠，即可得到一般积木堆叠问题的一个 2-近似解，如图 10 所示。图 6 中的例子表明，这一近似比是紧的。

定理 16. 无平衡配重的积木堆叠的最优解构成一般积木堆叠问题的一个 2-近似解。

证明。设 $p \in B$ 为突出块，且 $\pi\colon B \to \{1, \ldots, |B|\}$ 为（采用平衡配重的）最优堆叠配置的堆叠顺序。现在考虑一种完全右对齐堆叠，其中块 p 位于顶部，其余部分的堆叠顺序与原先相同。设 M_i、M_i' 分别为原堆叠顺序和新堆叠顺序对应的聚合质量，并注意，对于所有满足 $\pi(i) > \pi(p)$ 的 i，均有 $M_i = M_i'$。根据 (3)，新堆叠的悬伸量为

$$\begin{aligned}
\sum_{i \in B} \frac{w_i m_i}{M_i'} &= w_p + \sum_{i \in B \setminus \{p\}} \frac{w_i m_i}{M_i'} \\
&\geq w_p + \sum_{i\colon \pi(i) > \pi(p)} \frac{w_i m_i}{M_i'} \\
&\geq \frac{w_p}{2}\Big(2 - \frac{m_p}{M_p}\Big) + \frac{1}{2} \sum_{i\colon \pi(i) > \pi(p)} \frac{w_i m_i}{M_i'} \\
&= \frac{1}{2}\Big(w_p\Big(2 - \frac{m_p}{M_p}\Big) + \sum_{i\colon \pi(i) > \pi(p)} \frac{w_i m_i}{M_i}\Big),
\end{aligned}$$

也就是说，至少达到原始堆叠配置悬伸量的一半（参见式（2））。 □



(a) (b) (c)

图 10：（a）是具有最大悬伸量的堆叠配置。根据定理 16，（b）的悬伸量至少是（a）的一半。因此，对于最佳完全右对齐堆叠（c）也同样成立。

因此，无平衡配重的积木堆叠问题的多项式时间近似方案（见推论 10）自动构成一般积木堆叠问题的一个（$2 + \epsilon$）近似算法。

推论17。对于任意 $\epsilon > 0$，一般积木堆叠问题都存在一个（$2 + \epsilon$）近似算法。

8 结论。

我们介绍了积木堆叠问题的几个新颖方面。首先，我们发现，构造悬伸量尽可能大的堆叠是 NP 困难的。在

此外，我们发现，积木堆叠不仅仅是一个有趣的谜题：不使用配重块的叠堆与飞机加油问题以及现实中的预约调度难题密切相关。这一系列问题能否高效求解，仍是一个引人入胜的开放问题。尽管如此，这种联系使我们得以为不采用平衡配重的积木堆叠问题设计出一个多项式时间近似方案，并为一般情形设计出一个 $(2+\epsilon)$-近似算法。

参考文献

[1]Bauerhenne, C., Kolisch, R., Schulz, A. S.（2024年）。具有等待时间保证的鲁棒预约调度。arXiv。doi.org/10.48550/arXiv.2402.12561[2] Fine, N. J.（1947）。吉普车问题。《美国数学月刊》。54(1)：24-31。doi.org/10.2307/2304923[3] Franklin, J. N.（1960）。机队航程。《工业与应用数学学会杂志》。8(3)：541-548。doi.org/10.1137/0108039[4] Gale, D.（1970）。吉普车问题再探，或一打吉普车。《美国数学月刊》。77(5)：493-501。doi.org/10.1080/00029890.1970.11992525[5] Garey, M. R., Johnson, D. S.（1979）。《计算机与不可解性：NP 完全性理论指南》。旧金山：W. H. Freeman。[6] Gamzu, I., Segev, D.（2019）。飞机加油问题的多项式时间近似方案。《调度杂志》。22(1)：119-135。doi.org/10.1007/s10951-018-0569-x[7] Hall, J. F.（2005）。积木堆叠之趣。《美国物理学杂志》。73(12)：1107-1116。doi.org/10.1119/1.2074007[8] Hausrath, A., Jackson, B., Mitchem, J., Schmeichel, E.（1995）。Gale 的往返吉普车问题。《美国数学月刊》。102(4)：299-309。doi.org/10.1080/00029890.1995.11990575[9] H¨ohn, W., Jacobs, T.（2015）。非线性成本下单机调度中 Smith 规则的性能。《ACM 算法汇刊》。11(4)：1-30。doi.org/10.1145/2629652[10] Horton, G. K., Holton, B. E., Freidkin, E.（1997）。倾斜的瓦片塔：再探。《物理教师》。35(4)：214-219。doi.org/10.1119/1.2344654[11]Johnson, P. B.（1955）。Leaning Tower of Lire。《美国物理学杂志》。23：240。doi.org/10.1119/1.1933957[12] Kazachkov, A., Kireˇs, M.（2017）。用微积分重建纸牌堆。《物理教育》。52(4)：045019。doi.org/10.1088/1361-6552/aa6a4e

[13] Korf, R. E.（2022）穿越未知宽度沙漠的吉普车，《美国数学月刊》。129(5)：435-444。doi.org/10.1080/00029890.2022.2051404[14]Lengv´arszky, Z. 和 Shepherd, D.（2023）。黏性堆叠的最大悬挑。《数学杂志》。96(5)：540-550。doi.org/10.1080/0025570X.2023.2266324[15] Li, J., Hu, X., Luo, J., Cui, J.（2019）。飞机加油问题的快速精确算法。国际组合优化与应用会议。施普林格。第316-327页。doi.org/10.1007/978-3-030-36412- 025

[16]Megow, N., Verschae, J. (2018). 变速机器上调度的对偶技术。《SIAM 离散数学杂志》32(3): 1541-1571. doi.org/10.1137/16M105589X

[17] Mittal, S., Schulz, A. S., Stiller, S. (2014). 鲁棒预约调度。《近似、随机化与组合优化：算法与技术》（APPROX/RANDOM 2014）。达格施图尔城堡—莱布尼茨信息学中心。doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.356[18]Paterson, M., Peres, Y., Thorup, M., Winkler, P., Zwick, U. (2009). 最大悬挑。《美国数学月刊》。116(9)：763-787。doi.org/10.4169/000298909X474855[19] Paterson, M., Zwick, U. (2009). 悬伸量。《美国数学月刊》。116(1)：19-44。doi.org/10.1080/00029890.2009.11920907[20] Phipps, C. G. (1947). 吉普车问题：一种更一般的解法。《美国数学月刊》。54(8)：458-462。doi.org/10.1080/00029890.1947.11991865[21]Polster, B., Ross, M., Treeby, D. (2012). 一个连续悬挑的案例。《美国数学月刊》。119(2)：122-139。doi.org/10.4169/amer.math.monthly.119.02.122[22]Schulz, A. S., Udwani, R. (2019). 异质成本下的鲁棒预约调度。《近似、随机化与组合优化：算法与技术》（APPROX/RANDOM 2019）。达格施图尔城堡—莱布尼茨信息学中心。doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.25[23] Stiller, S., Wiese, A. (2010). 递增速度调度与流水车间调度。国际算法与计算研讨会。施普林格。第279-290页 doi.org/10.1007/978-3-642-17514-5 24[24] Treeby, D. (2018). 关于悖论塔的进一步思考。《美国数学月刊》。125(1)：44-60。doi.org/10.1080/00029890.2018.1390375

[25]V´asquez, O. C. (2015). 对于飞机加油问题，局部优先关系意味着全局优先关系。《优化快报》。9(4)：663-675。doi.org/10.1007/s11590-014-0758-2
[26] Woeginger, G. J. (2010). 飞机加油问题。研讨会 10071“调度”。达格施图尔城堡—莱布尼茨信息学中心。第24页。
doi.org/10.4230/DagSemProc.10071.3[27] Zhang, G. J., Cui, J. C. (2021). 一种新颖的混合整数线性规划模型，用于 N-车辆探索问题。《中国运筹学会学报》。9(2)：359-373。doi.org/10.1007/s40305-019-00289-2