arXiv:2603.08537v1 [cs.CG] 9 Mar 2026

并行滑动立方体

Hugo A. Akitaya
美国马萨诸塞大学洛厄尔分校 Miner计算机与信息科学学院
Joseph Dorfer
奥地利格拉茨工业大学算法与理论研究所

Peter Kramer
德国布伦瑞克工业大学计算机科学系

Christian Rieck
德国卡塞尔大学数学研究所

Gabriel Shahrouzi
美国马萨诸塞大学洛厄尔分校Miner计算机与信息科学学院

Frederick Stock
美国马萨诸塞大学洛厄尔分校Miner计算机与信息科学学院

摘要

我们研究三维中可编程物质在并行重构下的经典滑动立方体模型，不仅推广了已知的最佳界限（从二维推广到三维），还给出了新的算法结果和出人意料的复杂度结果。一般而言，该问题要求在满足连通性约束的情况下，为两个连通构型之间的 n 个不可区分的单位立方体模块求出重构序列；必须始终存在一个连通的骨干。重构序列的完工时间是执行的并行移动次数。

我们证明，判定是否存在这样的序列是 NP-困难的，即使完工时间为常数，且两个输入构型的对称差大小为常数也是如此，从而解决了 [3] 中的一个悬而未决问题。具体而言，判定最优完工时间是 1 还是 2 是 NP-困难的。我们还证明，该问题在顺序模型和并行模型中均具有对数 APX-困难性，强化了 [2] 中关于 APX-困难性的结论。最后，我们概述了一种渐近最坏情形最优的输入敏感重构算法。生成的序列长度取决于输入构型的包围盒，在最坏情形下，完工时间为$\mathcal{O}(n)$。

2012年 ACM 学科分类计算理论 → 计算几何

关键词与短语滑动方块、并行移动、可重构性、三维、常数
完工时间、log-APX困难性、NP困难性、最坏情况最优性

资助*Hugo A. Akitaya*、*Gabriel Shahrouzi* 和 *Frederick Stock*：由美国国家科学基金会（NSF）资助，项目编号 CCF-2348067。*Joseph Dorfer*：奥地利科学基金（FWF）10.55776/DOC183。*Peter Kramer*：德国学术交流中心（DAAD）奖学金。*Christian Rieck*：由德国研究联合会（DFG）资助，项目编号 522790373。

1 引言

可编程物质系统由大量能够自重构的简单模块组成，因其在机器人学、制造业和分布式计算中的应用而受到广泛关注。从计算几何的角度来看，一个基本的算法挑战是：在保持连通性并避免碰撞的同时，将一个连通的模块构型转换为另一个构型。Fitch、Butler 和 Rus [1, 2, 3, 4, 10, 11, 15, 16, 18, 19], 对此问题进行了深入研究 [14],：模块在方形（或超立方体）晶格上通过离散滑动或凸过渡进行移动，如图 1 所示。

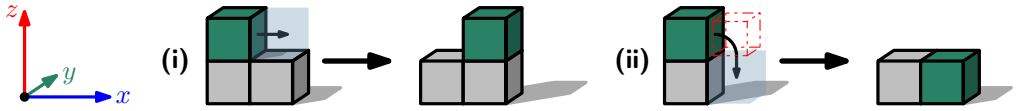


图 1 两种合法移动：(i) 滑动和 (ii) 凸过渡。

早期研究主要关注顺序模型，即每次只有一个模块移动。当实例为二维或三维时，我们分别将该模型称为顺序滑动正方形模型或顺序滑动立方体模型。Akitaya 等人 [2] 研究了顺序滑动正方形的输入敏感算法，为紧凑化一个包含 n 个模块的构型提供了 $\mathcal{O}(nP)$ 界，其中 P 为包围盒周长。他们的算法也是原地的，这意味着除一个模块外，所有模块始终都被限制在初始构型的包围盒内。他们还通过从平面单调 3-SAT 进行归约，证明了最大完成时间最小化问题的判定版本是 NP 完全的。所构造的实例是一对构型，其对称差的规模与变量数量呈线性关系。作者还声称，同一归约可以加以调整，以证明在三维实例中，对最小完工时间进行近似是 APX-困难的，但他们没有给出证明。

最近，Akitaya 等人 [3] 提出了二维并行滑动方块模型，指出与顺序调度相比，并行移动可以大幅缩短完工时间。他们的算法实现了最坏情形最优的完工时间 $\mathcal{O}(P)$，其中 P是包围盒的周长。他们证明，判定一次重构能否在完工时间为 1（无标签情形）或 2（有标签情形）内完成是 NP 完全的。这排除了以完工时间为参数的固定参数可处理算法。不过，与 [2], 一样，他们的归约产生了具有线性规模对称差的实例，因此作者提出了一个问题：如果输入的对称差规模为常数，最大完成时间最小化问题是否仍然是困难的?

对于三维实例，Abel 和 Kominers 首先证明了通用重构，随后他们与其他合作者将顺序移动次数的界改进至 $\mathcal{O}(n^2)$ [1] ，并使其算法能够原地运行。Kostitsyna 等人 [16]证明，将任意给定构型紧凑化所需的移动次数可以达到最优，且与输入中立方体坐标的坐标和成正比。迄今为止，在允许模块并行移动时，通用重构仍没有更好的界。

1.1 我们的贡献

我们将 [3] 提出的并行滑动正方形模型推广到三维（我们称之为并行滑动立方体），并得出了正面和负面结果。

- 并行滑动立方体中的最大完成时间最小化问题的判定版本是 NP-困难的，甚至难以区分完工时间为 1 和 2 的情形。我们的归约使初始构型与最终构型之间的对称差为 1，这意味着不存在固定参数可处理算法。并行滑动立方体和顺序滑动立方体中的最大完成时间最小化问题都是对数近似困难的，且无法实现优于 $\Theta(\log n)$ 的近似因子。对于最大完成时间为 $\mathcal{O}(A + h)$ 的通用重构，可以在多项式时间内计算调度，其中 A 是构型在其包围盒底面上的投影面积， h 是包围盒的高度。最坏情形下，这个输入敏感界变为 $\mathcal{O}(n)$，且该界是最坏情形最优的。

1.2 相关工作

由 Fitch、Butler 和 Rus [14], 提出的滑动立方体模型，已成为模块化重构算法研究中的核心抽象。在串行模型中，已有关于移动次数的二次最坏情况复杂度的通用算法（适用于正方形、立方体和超立方体） [1, 2, 10, 16]。 [1, 4] 中的原地算法采用脚手架技术（即一种规则的子结构，用于保证算法各步骤之间的连通性；该技术由 [17] 和 [20]分别独立提出），构造出一个扫描平面来压缩输入构型，与我们的算法类似。

并行重构日益受到关注。在二维分布式算法领域，Dumitrescu 等人 [11] 首次研究了这一问题，并为两个 y-单调形状之间的重构提供了线性总完成时间。Michail 等人 [18] 和 Wolters [23] 也提出了适用于特定输入的算法。Hurtado 等人 [15] 证明，限制较少的模型能够在实现普适性的同时达到线性总完成时间。他们还展示了如何利用元模块——由多个协作模块组成、行为如同单一实体的小型群组——在滑动正方形模型中，对已由元模块构成的构型复现相同结果。Akitaya 等人 [3] 采用元模块和支架结构技术（与 [1, 4]类似），为一般输入设计了最坏情况最优算法。虽然我们的算法大体遵循相同原则，但将这些思路推广到三维情形极具挑战性，因此需要若干新的结构性证明。

近似。 算法重构领域的大多数文献都关注最坏情况界，因为获得最优解通常更具挑战性（而且在计算上更常常不可处理）。一个典型例子是凸点集三角剖分之间的翻转距离。尽管最优的最坏情况界已知，但最小化问题的复杂度仍是一个著名的开放问题 [22]。与我们的问题更为相关的是，Demaine 等人 [8] 为机器人群体的并行运动规划获得了常数因子近似，其中与网格对齐的正方形可以移动到相邻的网格单元（类似于我们的滑动移动），但不受连通性约束。他们还证明，除非 P=NP，否则该问题不允许完全多项式时间近似方案，但其 APX-困难性尚未得到证明。在串行设置下，该问题已被证明是 APX-困难的 [6]。

使用类似的技术， [8], Fekete 等人 [12, 13]在要求连通性的情况下实现了常数伸长率，即完工时间在单个机器人必须在输入构型之间移动的最大最小距离的常数因子以内，但所采用的并行移动模型与我们的模型不同（也更为宽松）。然而，他们的算法只能应用于缩放构型，因此不适用于一般输入。

如前所述，Akitaya 等人 [2] 声称，在顺序滑动立方体模型中，最小化完工时间是 APX-困难的。我们的近似困难性结果是针对并行滑动立方体模型的首批结果；据我们所知，这也是此类运动规划重构问题中首个对数 APX-困难性结果。

2 预备知识

我们的研究考虑单位立方体在三维整数网格中的协同运动，沿用并扩展了 [3] 中的记号。对于 $v \in \mathbb{Z}^3$，令 $x(v)$ $y(v)$ $z(v) \in \mathbb{Z}$ 表示其沿三个坐标轴的坐标。这三个坐标轴依次指向东、北和上 。为了测量整数网格中的距离，我们使用如下定义的 L_1 和 L_∞ 范数：

$$\|u\|_1 = |x(u)| + |y(u)| + |z(u)| \qquad \text{and} \qquad \|u\|_\infty = \max\{|x(u)|, |y(u)|, |z(u)|\}.$$

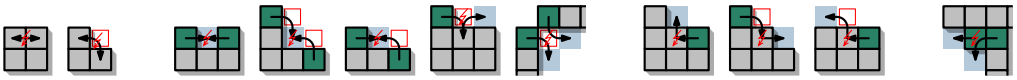


图 2 碰撞示例。图片引自 [3], , 经作者许可使用。

在我们的设定中，单元格 c 是一个立方体$\mathcal{Q}_3$, ，其锚定在其最小坐标处，记作 $c \in \mathbb{Z}^3$。因此，每个单元格有八个顶点和六个面，每个面都由一个四环围成。若两个单元格共享一个面、边或顶点，则它们分别是面-、边-或顶点相邻 的。

构型。 立方体模块 的构型 C 定义为由 n 个单元格构成的集合，每个单元格中恰好占据一个模块；当且仅当被占据单元格的面邻接关系（ C的对偶图）所诱导的图是连通图时，该构型才有效。对于给定的 C，我们将包含所有模块的最小轴对齐包围盒 记为 B。包围盒沿 x、 y和 z 坐标轴的延展范围 分别定义其宽度、深度 和高度 （请回忆， z 为上）。

移动。 模块可以执行两种移动：滑动 和凸过渡。滑动是指模块沿两个相邻模块的表面平移一个单位，而凸过渡则是指模块沿单个模块的表面跨越两个坐标轴移动。按照 [3], 的记号和约定，这两种移动都耗时一个单位，并以恒定速度执行。

变换。 并行移动以离散的变换形式执行，每次持续一个时间单位。这些变换必须保持骨干连通，因此移动的模块必须是可自由移动的：移动中的模块不提供连通性。若从构型 C 中移除集合 M 的任意子集后，所得构型仍然连通，则称模块集合 M 在构型 C 中是可自由移动的。于是，若集合 $\{\mathrm{m}\}$ 是可自由移动的，则单个模块 $\mathrm{m} \in C$ 也是可自由移动的。此外，变换不得导致碰撞，即两个模块的体积发生重叠。例如，这禁止模块在变换过程中交换位置、进入同一个单元格，或经过同一个单元格进行凸过渡。详情请参阅 [3] 和图 2。

问题定义。 并行滑动立方体问题给定两个各含 n 个模块的构型以及一个整数 $k \in \mathbb{N}^+$，要求找到一个至多包含 k 次变换的序列，使一个构型变为另一个构型。这样的序列称为完工时间为 k 的调度。

至此，我们已完整描述模型；以下介绍辅助术语和记号。

投影。 我们将构型 $C \in \mathbb{Z}^3$ 沿坐标轴 $\lambda \in \{x, y, z\}$的投影 定义为二维构型 $\lambda[C] = \{(\rho(\mathrm{m}), \phi(\mathrm{m})) \mid \mathrm{m} \in C, \{\rho, \phi\} = \{x, y, z\} \setminus \{\lambda\}\}$。此外，我们将二维投影的面积 定义为 $\mathbb{A}(\lambda[C]) = |\lambda[C]|$。

邻域。 本文中我们始终使用单元格和模块的邻域 概念。设 $c \in \mathbb{Z}^3$。我们将 c 的开 L_1 邻域和 L_∞ 邻域定义为

$$N(c) \coloneqq \{c' \in \mathbb{Z}^3 \mid \|c - c'\|_1 = 1\} \qquad \text{and} \qquad N[c] \coloneqq \{c' \in \mathbb{Z}^3 \mid \|c - c'\|_\infty = 1\}.$$

例如，通过添加星号 $N^*[c]$，我们表示也包含 c的邻域。稍微滥用一下记号，这些定义可以推广到单元格集合 $C \subset \mathbb{Z}^3$，例如，

$$N^*(C) \coloneqq \bigcup_{c \in C} N^*(c) \qquad \text{and} \qquad N(C) \coloneqq N^*(C) \setminus C.$$

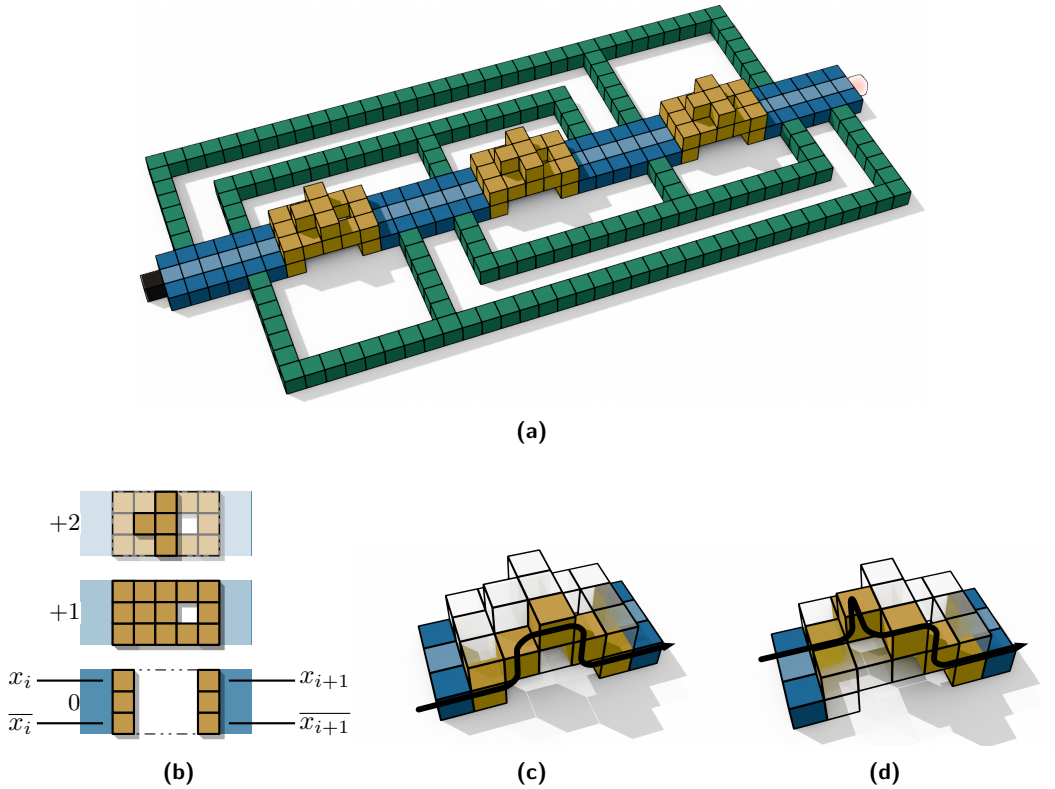
（闭） r邻域是第 r 次迭代，例如，$N_r^*(c) = N(N_{r-1}^*(c))$，其中 $N_1^* = N^*$。

3 计算复杂性

本节给出并行滑动立方体的若干复杂度结果。首先，我们通过证明三维情形下完工时间为 1 且对称差为 1 时为 NP-困难，否定了 [3] 中提出的一个问题。这表明，以对称差为参数的并行滑动立方体问题并非固定参数可处理。随后，我们通过从集合覆盖归约，证明其具有对数 APX-困难性；该归约本质上也适用于串行设置。我们先证明对称差为常数时的 NP-困难性。

▶**定理 1** (⋆)。 对于完工时间为 NP-完全，且对称差大小为 1 的并行滑动立方体问题，1。

证明概要。 我们从 平面单调 3-SAT进行归约，该问题要求判断给定的布尔公式是否可满足 [7]。每个子句至多包含 3 个文字，且要么全部为正文字，要么全部为负文字；此外，子句-变量关联图必须存在一种平面嵌入，使变量映射到 x轴上，正（分别为负）子句映射到上（分别为下）半平面，并且边不与 x轴相交。对于该构造，我们根据一个公式放置多组共三种组件类型 φ ，如图 3(a)所示。



(a) (b) (c) (d)

图3 (a) 展示了我们的构造，颜色表示文中所述的各个构件，其中 $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4)$。变量构件从左至右按升序排列。(b)–(d) 展示了连接构件及可能经过的路径。

6 并行滑动立方体

具体而言，我们沿 x轴、在 xy平面内放置变量构件（蓝色），并根据其y坐标区分变量构件中的正模块和负模块。在任意两个相邻变量之间，我们放置一个连接构件（黄色）。详细分解见图 3(b)。此外，每个变量构件至少连接一个子句构件（绿色）；该构件实际上是一种细长的梳状结构，连接到其中每个变量构件的正侧或负侧。最后，在构造的一端放置一个差异模块（黑色/透明），使其处于初始配置；在目标配置中，则位于构造的另一端。它包含整个对称差。为了在同一步中清空差异模块的初始单元格并填充其目标单元格，我们必须找到一条从一个位置通往另一个位置、且不会将构型分割开的路径。具体而言，该路径会经过每个变量构件的正侧或负侧，将其与子句分隔开来，从而编码一个布尔值赋值。
◀

▶**推论 2** ($\star$). 除非 P$=$ NP，否则即使对于常数大小的对称差，并行滑动立方体也无法在多项式时间内以优于 2 的因子进行近似。

我们强调，我们仅出于清晰起见而从平面单调 3-SAT 进行归约。尽管如此，同一构造在从可满足性进行归约时仍然有效；当子句-变量关联图非平面时，子句可以嵌入第三维。以下是定理 1 的直接推论。

▶**推论 3.** 并行滑动立方体在以对称差的大小为参数时不是 固定参数可处理 的。

接下来，我们证明滑动立方体的并行变体与串行变体均为对数近似困难。我们给出一个从集合覆盖出发的归约。我们的整个构造可置于大小为 $\mathcal{O}(n^4k^7)$ 的包围立方体之内，其中 n, k 指具有 n 个元素和 k 个集合的集合覆盖实例。其对大型三维结构的依赖使其不适用于该问题的二维版本。

▶**定理 4** ($\star$)。 除非 P $=$NP，否则并行滑动立方体和顺序滑动立方体都是对数近似难的；因此，它们无法以优于 $\Theta(\log(n))$ 的因子进行近似。

证明概要。 我们将 对数近似难度证明分为两部分，研究对象是 并行滑动立方体 。第一部分中，我们证明一个更易分析的问题变体是 对数近似难的，我们称之为 不可移动。在这个变体中，模块分为两类：(1) 不可移动模块 不能执行任何移动，因此始终处于同一位置；(2) 可移动模块 则可以执行所引入的移动。请注意，不可移动模块仍可能造成碰撞，也可以提供骨干。此外，一次只有一个模块移动，这与我们在串行设置中的惯例一致。

第二部分中，我们论证，只要为每个不可移动模块增添一个尖刺构件，不可移动变体中的假设仍然成立。尖刺构件是一条沿固定方向延伸、长度很长且宽度为1的模块链，其唯一作用是阻挡所有在不可移动变体中不可移动的模块。一旦试图滑动这些模块，就会使尖刺构件与构造的其余部分断开，从而违反骨干约束。

我们的归约来自对数近似困难问题集合覆盖 [9]。一个实例$\Sigma_{n,\ k}$由自然数 1、……、 n 以及一个由$\{s_1$、……、 $s_k\}$ 组成的子集集合构成，这些子集取自$\{1$、……、 $n\}$，且满足 $\bigcup_{i=1}^{k} s_i = \{1$、……、 $n\}$。因此，任务是找出一个最小基数的集合 $S^\star$，使得$\bigcup_{s_i \in S^\star} s_i = \{1$、……、 $n\}$。图4展示了该构造蓝图的俯视图。

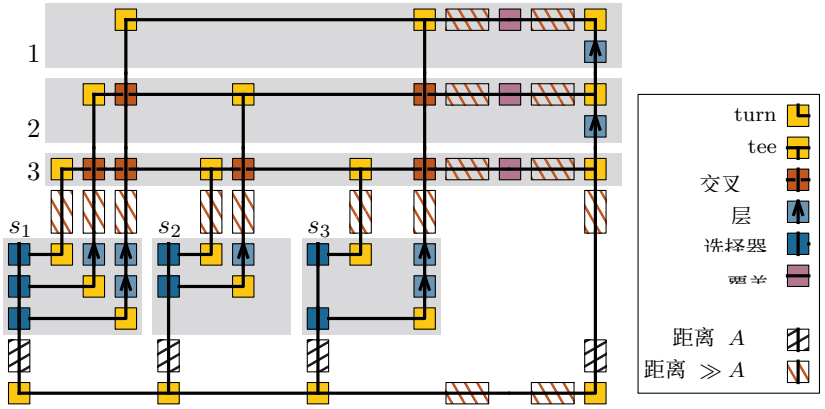


图4 展示了我们如何从集合覆盖实例归约而来，该实例包含 $n = 3$ 以及集合 $s_1 = \{1、2、3\}$、$s_2 = \{2、3\}$ 和 $s_3 = \{1、3\}$。

黑线表示由不可移动模块组成的细路径，每个模块的宽度和高度均为 1，这些路径将所有构件彼此连接起来。

我们引入几种不同的构件。与集合覆盖实例直接相关的构件是覆盖构件 和选择器构件。具体而言，对于每个$j = 1$，……，$n \in \mathbb{N}$，我们都会引入一个覆盖构件，在图4中以紫色方块表示，并在图5中详细展示。在覆盖构件中，不可移动模块路径被一个可移动模块 m$_j$打断，该模块在图5中以橙色突出显示。覆盖构件包含初始构型与最终构型的对称差中的模块：更确切地说，在 C_I中，橙色模块位于第0层，而在 C_F 中，它被抬升至第 $+1$层，恰好位于其原始位置正上方。为了使橙色模块能够在覆盖构件内移动，我们必须确保骨干保持连通。这通过选择器构件实现；选择器构件在图4中以深蓝色方块表示，并在图6中详细展示。该构件用于表示一个数 j 属于集合 s_i。在选择器构件中，一条不可移动模块路径与另一条路径正交排列，且两条路径都连接到与数 j相关联的覆盖构件，但连接方向不同。移动一个可移动模块以连接这两条路径，即可建立骨干约束，进而使覆盖构件中的橙色模块得以移动。其余构件用于处理连通性、交叉，以及为实现其中一些交叉所需的路径抬升。

该归约的工作方式如下。由于 C_I 的对偶图构成一棵树，对称差中的任何橙色模块都无法在不违反骨干约束的情况下移动。我们从 C_I中的单个可移动模块 m 开始，将其放置在左下角，并使其朝选择器 s_i 移动，以填补其内部的空隙。这样一来，我们在对偶图中闭合一个环，从而使属于集合 s_i 的覆盖构件内部的那些橙色模块能够移动到各自的最终位置。由于底行（它与选择器构件形成梳状结构，迫使我们必须经由它进行路由）与任何选择器之间的距离很大，我们力求尽可能少地穿越这些距离。

尚需证明的是，集合覆盖实例的最优解与最小完工时间的调度之间的等价性。这源于对构件之间距离的适当调整，以及对所有可行移动的详尽分析。◀

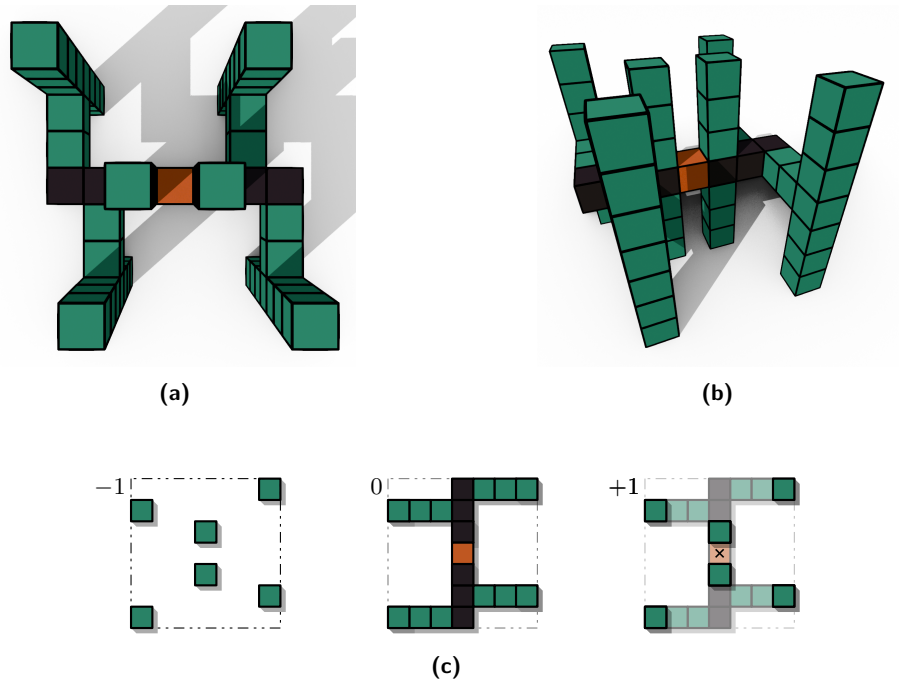


图 5 覆盖构件。橙色立方体表示对称差中的模块，其起始位置位于第 0 层，目标位置位于第 $+1$ 层。深色立方体表示不可移动模块。绿色立方体对应尖刺。

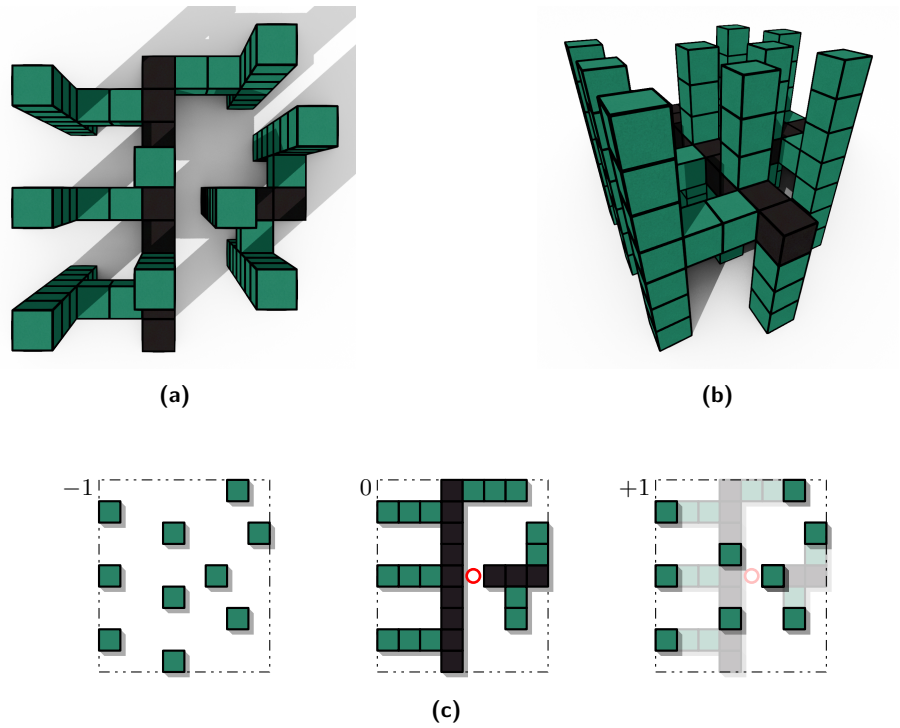


图 6 选择器构件使覆盖构件内的模块能够执行移动。

4 传送

给定构型 C 的一个子集 Q，如果某个调度 传送 Q，即将 C 变换为一个新构型 C'，且满足 $(C \setminus Q) \subset C'$，我们就称该调度如此。我们证明，给定两个构型 C 和 C'，使得 $C' = (C \setminus S) \cup S'$，其中 $C \setminus S$、S 和 S' 都是连通子构型，则存在一个调度将 C 变换为 C'，也就是说，S 传送为 S'。这一结果本身也具有研究价值，但本文不尝试最小化此类变换的完工时间。为简化起见，我们没有这样做，因为我们的主要结果只使用这些调度来处理大小恒定的子构型，因此完工时间始终为常数。

本节的主要技术工具是对 [1] 中 LocateAndFree 的推广。LocateAndFree 会在构型的外表面定位一个模块，并在不改变外表面的情况下释放该模块。我们的要求更严格：我们希望在传送的同时，保持构型的其余部分（内部和外部）不变。

构型 C 的一个面，是补集 C （ $\mathbb{Z}^3$ 中不属于 C的单元格集合）某个连通分量边界上的模块面集合。 C 的外表面，是 C无界连通分量边界上的面。若 C 某个面中的两个相邻模块面 f_1 和 f_2 满足以下任一条件，则称它们滑动相邻：它们围成同一个单元格 $c \in C$；或者，在构型 $C \cup \{c_1\}$中，包含 f_1 的模块 $c_1 \in C$经过一次移动后可以到达包含 f_2 的单元格 $c_2 \in C$。 C 的每个面都可以看作一个二维流形，它由沿公共边合并滑动相邻的模块面而成。（更多详情请参见 [1] 。 C 的每个面都对应于 [1]中“滑动邻接图”的一个连通分量。）若构型面 C中的一条边 e 包含在该面内两个以上的模块面中，则称这条边为夹边。特别是在三维中，一条夹边恰好包含四个模块面，且至多属于两个构型面。当两个不面相邻的模块共用一条边时，就会出现这种情况。我们将 C的扩展面定义为这样一种流形：在每个夹边处进行拓扑手术，将该夹边处的模块面两两连接，使得 f_1 和 f_2 位于 C中同一个模块的边界上。请注意， C 的一个扩展面可能包含 C 的多个面，因为拓扑手术可能会合并 C 的不同面。 C 的外部扩展面，是包含 C外表面的扩展面。若某个模块的一个面包含在 F 内，则称该模块位于 C 的面 F 中。直观地说， C 有且仅有一个扩展面，当且仅当其内部是单连通的：与 C 等价的多立方体的内部同胚于开球。

▶**引理5** ($\star$)。 任意连通构型的外部扩展面中都存在一个自由模块。

▶ **引理6** ($\star$)。 给定构型 C、连通子集 $S \subset C$，其中 $|S| \geq 2$ 使得$C \setminus S$ 连通；给定与 S相邻的空单元格 e，以及自由模块 $m \in S$，使得 e 与 m 位于 $S \setminus \{m\}$的同一扩展面中，并且 $(S \setminus \{m\}) \cup \{e\}$ 和 $(C \setminus \{m\}) \cup \{e\}$ 均连通。则存在一个调度，以完工时间 $2^{\mathcal{O}(|S|)}$ 将 m 传送至 e。

我们要求 $|S| \geq 2$ 和 $(S \setminus \{\mathrm{m}\}) \cup \{e\}$ 连通，以便将完工时间界定为 $|S|$的函数。如果这两个条件中任意一个不满足，则存在一些示例，要求 m 在不属于 S的模块上移动，从而使运行时间依赖于 $|C|$。如果我们只关注此类调度的存在性，则可以省略这些条件。给定两个构型 $C \cup S$和 $C \cup S'$，我们可以连续应用引理6，通过填充与 $S \cap S'$ 的一个连通分量相邻的 $S' \setminus S$ 中的空单元格，来扩展该连通分量。于是可以得到以下结果：

▶ **推论 7**。 给定两个构型 $C \cup S$ 和 $C \cup S'$，若其 C、S 和 S' 连通，则存在一种调度，可将 S 传送到 S'。

5 最坏情况最优算法

我们利用第4节的结果，在并行滑动立方体模型中高效地进行重新配置。从整体上看，我们的方法将 [3] 推广到了三维。整个策略分为三个阶段：聚集、压缩和紧凑重配置。聚集阶段的目标是构造一种称为蛇形结构 的结构，直观来说，它是一组相互连通的 $5 \times 5 \times 5$ 元模块。直观地说，蛇形结构可以在构型中自由、快速地移动，而不会影响其余构型的连通性。在压缩阶段，我们利用蛇形结构将构型粗略地投影到构型包围盒的某个面上。不失一般性，假设该面为顶面。接着，我们利用这一结构扫过包围盒，将构型的其余部分压缩成由 $5 \times 5 \times 5$ 个元模块组成的紧凑 构型。我们可以将前述阶段分别应用于 C_1 和 C_2，这样剩下的问题就归结为在两个紧凑构型之间进行重构，这由紧凑重配置阶段处理。按照文献中的定义，如果对于每个模块 $\mathrm{u} \in C$，由 u的位置与原点所确定的长方体完全包含在构型中，则构型 C 是紧凑的：

$$\forall \mathrm{u} \in C: \quad \forall a \in [0, x(u)]: \quad \forall b \in [0, y(u)]: \quad \forall c \in [0, z(u)]: \quad (a, b, c) \in C.$$

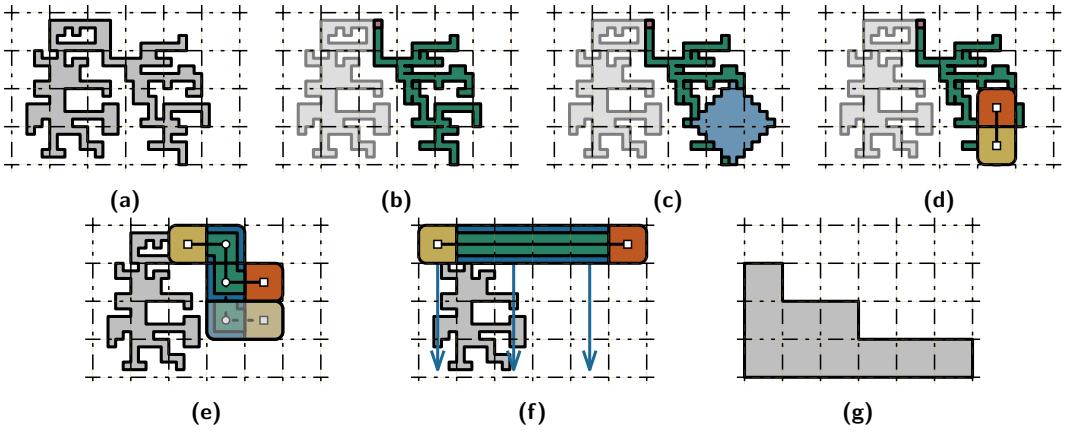


图7 我们的算法概览（未按比例绘制）。在 (a)–(d) 中，我们构造一个蛇形结构，并在 (e) 中将其扩展至 $\Theta(A + h)$ 个模块，再用它在 (f)–(g) 中压缩剩余构型。

为了指导 $5\times5\times5$元模块的创建和操作，我们将空间细分为由 $5^3 = 125$ 单位网格单元构成的 $5\times5\times5$元胞 。不失一般性地，我们可以假设输入构型 B_i 的初始包围盒 C_i, $i \in \{1,2\}$的各个维度均为 5 的倍数；否则，我们假设 B_i 为包含 C_i的最小此类包围盒（每个维度最多增加 4 个单位）。

由于前两个阶段处理的是单个输入构型，因此现在假设给定一个构型 C，其包围盒为 B。我们使用元胞分解来定义该构型的生成树 T，以引导收集阶段。令 $\mathcal{C}$ 为由每个元胞诱导出的 C 的连通分量集合：限制在单个元胞内的 C 可能是不连通的，这些连通分量中的每一个都是 $\mathcal{C}$ 的一个元素。我们按以下方式构建 T，使其成为 C 的生成树。计算 $\mathcal{T}$，即 $\mathcal{C}$ 的邻接图的一棵生成树，并任意选定其根节点。以 $\mathcal{T}$ 为指引，我们按如下方式构建 T。连通分量 $S \in \mathcal{C}$ 的根节点，要么是 S 中任意一个模块（如果 S 是 $\mathcal{T}$的根节点），要么是 S 中一个与

其父连通分量相邻的模块（该父连通分量属于 $\mathcal{T}$）。从每个连通分量 $\mathcal{C}$ 的根节点出发，计算其深度优先搜索生成树，使 T 成为生成森林。随后连接相邻的树，直到得到 C 的一棵生成树；连接时，将某个连通分量的根节点连接到其在 $\mathcal{T}$中的父连通分量。T 的根节点就是其根连通分量在$\mathcal{C}$中的根节点。

在压缩阶段，我们需要在蛇形结构中聚集至少 $(A+h)$125 个模块。为了实现 $\mathcal{O}(A+h)$ 的完工时间，收集阶段最多只能聚集常数倍的 $(A+h)$ 个模块。因此，我们需要找到 T_{s} 的一棵大小为 $\Theta(A+h)$ 的合适子树，并将其转换为蛇形结构。详情见 C.1节。

5.1 收集阶段

5.1.1 初始化

创建蛇形结构的第一步，是获得一个被模块完全填满的元胞。本节专门讨论这一任务。首先，我们使用引理 22，得到一棵大小为 $|T_{\mathsf{m}}| \geq 2625$ 且 $|T_{\mathsf{m}}| \in \mathcal{O}(1)$ 的子树 T_{m}。2625 这个数字来自半径为 12 的 L_1 球中所包含的单元格数量。简单来说，我们的策略是逐个使用 T_{m} 的模块，以模块 m 为中心扩展一个球（使用引理6）。如果该球的半径达到 12，就可以保证其中包含一个蛇形子配置。虽然二维版本的这一结果更容易实现（ [3]中就得到了类似结果），特别是有了模块传送这一新技术工具，但三维版本需要进行相当复杂的分类讨论，即使充分运用引理6的全部威力也是如此。这是因为在三维中，构型的其余部分（$C \setminus T_{\mathsf{m}}$）有更多方式阻挡用于填满目标球的模块前进。

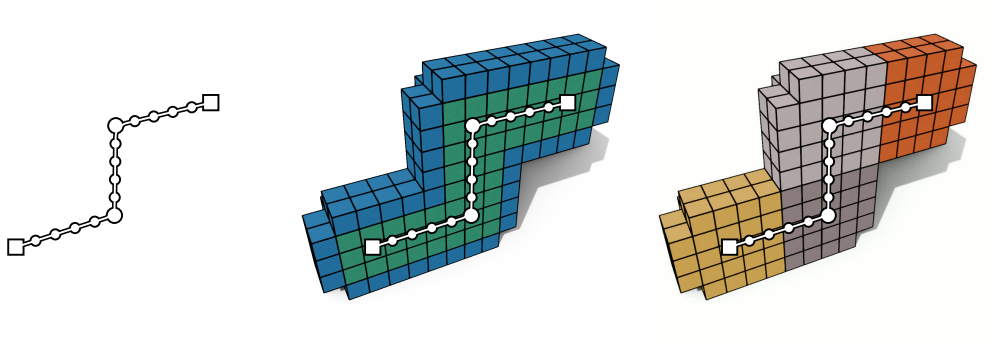
▶ **引理8** $(\star)$。给定 T 的一个大小为 $|T_m| \geq 2625$ 且 $|T_m| \in$为 $O(1)$的子树 T_m，存在一个 $O(1)$ 调度，可生成一种构型，使得包含 m 的元胞为满，并且 $T \setminus T_m$ 中的每个单元格都保持为满。

5.1.2 寻找蛇形结构

现在我们将精确描述蛇形结构及其能力，并说明如何构造它。

脊柱路径。 长度为 ℓ 的脊柱路径 $P \in \mathbb{Z}^{3\times\ell}$ 是一条简单的诱导路径，位于单元格的面邻接关系中，由三类顶点组成：端点（头部 和尾部）、主顶点和次顶点。头顶点通过一条最多包含 9 个次顶点的诱导路径与第一个主顶点相连。每两个相邻的主顶点之间由一条包含 4 个次顶点的直线轴平行路径相连。最后一个主顶点再通过另一条最多包含 9 个次顶点的诱导路径与尾顶点相连。为避免自交，沿路径相距 ≥ 5 的顶点之间还必须相距 ≥ 5个 L_1。

单元空间。 令 P 为一条脊柱路径。脊柱为 P 的蛇形结构的内部单元随后被定义为 P的闭 L_∞-邻域 $N^*[P]$，而外壳单元则是内部单元的开 L_1-邻域， $N(N^*[P])$。它们的并集构成蛇形结构的单元空间 $S(P) \subset N_4^*(P) \subset \mathbb{Z}^3$。我们根据主顶点划分单元空间：对于一个不是端点的主顶点 $v \in P$，其区段恰好为 $N_2^*[v] \cap S(P)$，其余所有单元则分别归入头部区段或尾部区段。这如图 8 所示。

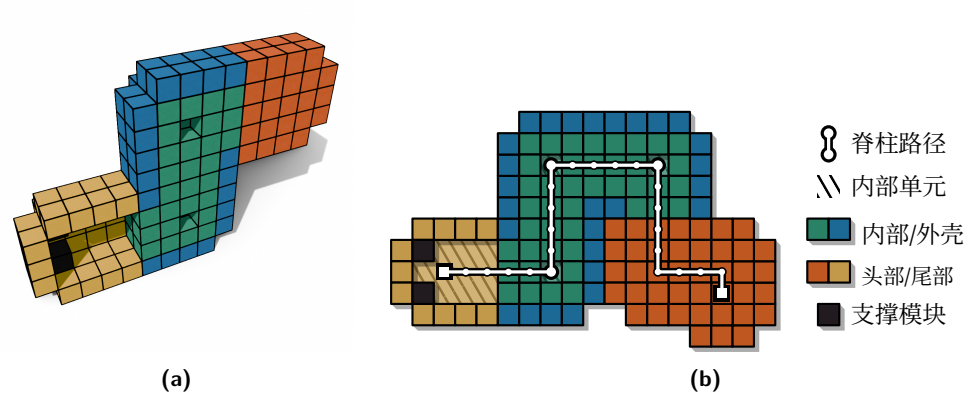


(a) (b) (c)

图 8（a）中所示长度为 16 的脊柱所定义的单元空间。为清晰起见，仅显示了一个 ha lf 。在（b）中，外壳和内部以不同颜色表示，（c）展示了各区段。

蛇形配置。一个蛇形（子）配置是由脊柱路径的单元空间 $S(P) \cap C$ 诱导出的（子）配置，其中除以下单元外的所有单元均被占据：(S.1) 脊柱发生弯折处的主顶点。(S.2) 尾部区段内部的任意子集，但与三个外壳单元相邻的内部单元除外；这些内部单元必须由支撑模块占据。

由于蛇形结构占据了其单元空间的大部分，按此定义的蛇形结构会形成一个连通子构型。此外，将脊柱弯折处的主顶点留空，可以在任意给定蛇形结构的头部区段和尾部区段之间高效传送模块。



(a) (b)

图 9 蛇形配置的结构。（a）有效蛇形构型的截面。（b）为清晰起见，采用颜色标记进行的二维示意图。

本节余下部分假设我们已经有一个蛇形配置。一旦 C由于引理8而包含一个足够大的 L_1 球，就总能找到一个常数大小蛇。

▶**引理 9**。如果一个构型包含一个完全占据的 L_1 球，其半径为12，那么它包含一个自由蛇形构型，其脊柱路径长度为11 ，且有198个模块。

作为关键的第一步，我们定义推操作及其逆操作拉。推操作可用于修改现有的蛇形配置，即移动头顶点并将脊柱路径延长一个单位。类似地，拉操作通过使尾顶点沿脊柱路径向头部移动，将脊柱路径缩短一个单位。

推。推操作可在尾部区段内部至少有16个模块的蛇形配置上执行。它沿给定方向移动头部，并将脊柱路径延长一个单位，从而在所得单元空间中形成有效蛇形构型。为填充新的外壳单元，它将最多12个尾部内部模块传送到头部。

▶**引理 10** ($\star$)。 推操作可以通过完工时间为 $O(1)$ 的调度来执行。

我们将推定义为一种朝前的操作，用于扩展蛇形结构；需要指出的是，通过传送，蛇形结构中头部和尾部的作用实际上可以互换。

▶ **引理 11** ($\star$)。 给定一个蛇形配置 $S(P)$，头顶点和尾顶点的作用可以通过 $O(1)$ 次变换互换。

每次推操作都会扩展蛇形结构的单元空间，可能会包含额外的模块。其中一些模块最初可能并非空闲，这意味着移动蛇形结构时，我们需要考虑可能存在的连通性约束。我们区分蛇形结构的 已拥有 单元格和 持有 单元格及模块，并对它们进行动态跟踪和更新。具体而言，如果某个单元格包含一个非空闲模块，而该单元格是推操作的结果，那么我们称蛇形配置 持有 该单元格；该模块也会被持有。一次推操作会将另外 21 个单元格纳入单元空间，其中某个子集必须被持有。另一方面，如果此类单元格在推操作前包含一个空闲模块，蛇形结构就可以取得该模块的所有权 。至关重要的是，根据引理 9，初始蛇形结构拥有其包含的全部 198 个模块。除非另有说明，蛇形结构会持有其移入的所有单元格，并会因以下操作而在移出时留下模块。

拉。 拉操作 会沿给定方向将尾部移动一个单位，从而缩短脊柱路径。如果这导致一个占用的单元格从单元空间中移除，蛇形结构会在该单元格中留下一个模块；因此，占用的模块数始终等于占用的单元格数。如果蛇形结构离开一个未占用的单元格，则不会留下模块，因此拉操作不会导致已拥有的模块数减少。这恰好与引理 10 相反：

▶**推论 12.** 给定一个尾部拥有足够多空的内部单元的蛇形子配置，可以通过完工时间为 $O(1)$ 的调度来执行拉操作。

我们定义了另外两种高层操作：分叉和连接。两者都严格依赖于沿蛇形结构脊柱推动、拉动和传送模块。

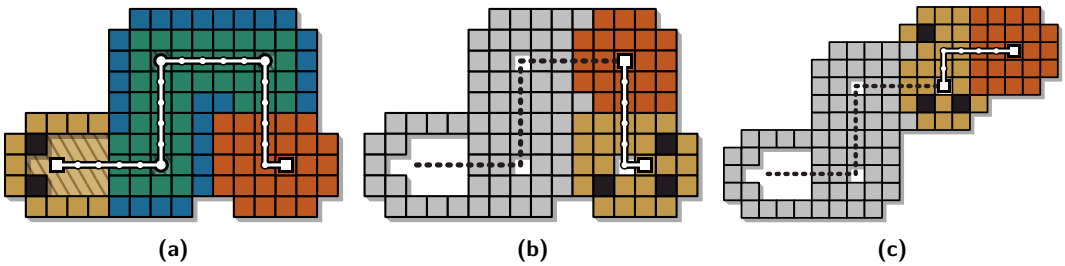


图 10 分叉与连接的示意图。在 (b) 中，一个主顶点成为新蛇形结构的头顶点。如果它们的脊柱路径的并集本身构成一条有效的脊柱，如 (c) 所示，则两条蛇形结构可以合并。

分叉。 分叉操作会以任意方向选取脊柱路径的一个前缀或后缀，并沿脊柱传送常数个内部模块，从而沿该脊柱构成一个头顶点已选定的有效蛇形构型，如图 10 所示。根据引理 11，这只需进行 $\mathcal{O}(1)$ 次变换。随后，新蛇形结构即可独立移动。

合并。 如果它们脊柱的并集构成一条有效脊柱，两条蛇便可在公共端点处合并，形成一条更大的蛇形结构。这是分叉的逆操作，见图 10。

这为我们提供了一种针对蛇形结构的通用重构方法，只要它与其余模块形成连通构型；同时也提供了基本工具，可通过取得额外模块的所有权，将初始蛇形结构生长为更大的结构。

5.2 生长蛇形结构

回想一下，我们的目标是使新确定的蛇形结构拥有 $\Theta(A+h)$ 个模块，即以模块 $s \in C$ 为根节点的树 $T_s \subseteq \mathcal{T}$ 中的所有模块。在本节中，我们将概述一种深度优先搜索方法，用于遍历该树并取得所有模块的所有权。

根据引理 9 确定的蛇形配置 $S(P)$ 完全包含在一个半径为 12 的 L_1球内，该球与子树 T_s 的模块相邻。我们首先 (i) 移动蛇形结构，使其与树所基于的 $5 \times 5 \times 5$ 网格的元胞对齐；然后再 (ii) 将其头部移至与 s 相同的元胞中。这两个步骤都可以仅通过推拉操作完成；沿着从其初始位置 T_s 到根节点的任意路径，所需的变换次数为 $\mathcal{O}(|T_s|)$。

在说明蛇形结构可以沿树移动之前，我们先证明它能够取得靠近其主顶点的空闲模块的所有权；图 11 给出了示意图。

▶**引理 13** ($\star$)。 设 S 是一个蛇形配置，其脊柱路径为 P，且尾部区段中有一个空单元格；设 $v \in P$是脊柱的一个主顶点。通过 $O(1)$ 次变换，可以将一个空闲模块 $m \in N_2^*[v]$传送到其内部，从而使蛇形结构取得该模块的所有权。

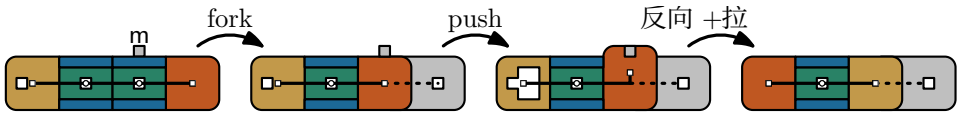


图 11 通过暂时分叉，蛇形结构可以取得邻近空闲模块的所有权。

根据引理 13，如果我们以五的倍数执行推拉操作，使蛇形结构的主顶点与网格的元胞对齐，那么只需让蛇形结构的任一主顶点与它需要取得所有权的模块位于同一元胞中即可。我们将证明，除推拉操作外，再使用合并与分叉操作，蛇形结构就能通过深度优先遍历树并取得叶节点的所有权，在 T_s 个 $\mathcal{O}(|T_s|)$ 变换内取得所有模块的所有权。

▶ **引理 14** ($\star$). 设 $m_1, \ldots, m_\ell$ 是 T_s 按 *DFS* 顺序排列的模块。包含 m_1 的蛇形配置 S 可以在 $O(\ell)$ 次变换内取得整个序列的所有权。

总体思路是让蛇形结构移动到树的一个叶节点，并在转身返回根节点 s 之前取得该节点的所有权。于是，序列 m_1、……、m_ℓ 可以看作沿着带有加倍边的树 T_s 走出的欧拉回路。每当当前蛇形结构无法仅通过反向移动来转弯时（也就是说，如果它的尾部沿树向上延伸得比其头部所需的位置更远），它就会暂时分叉。因此，我们需要跟踪一棵

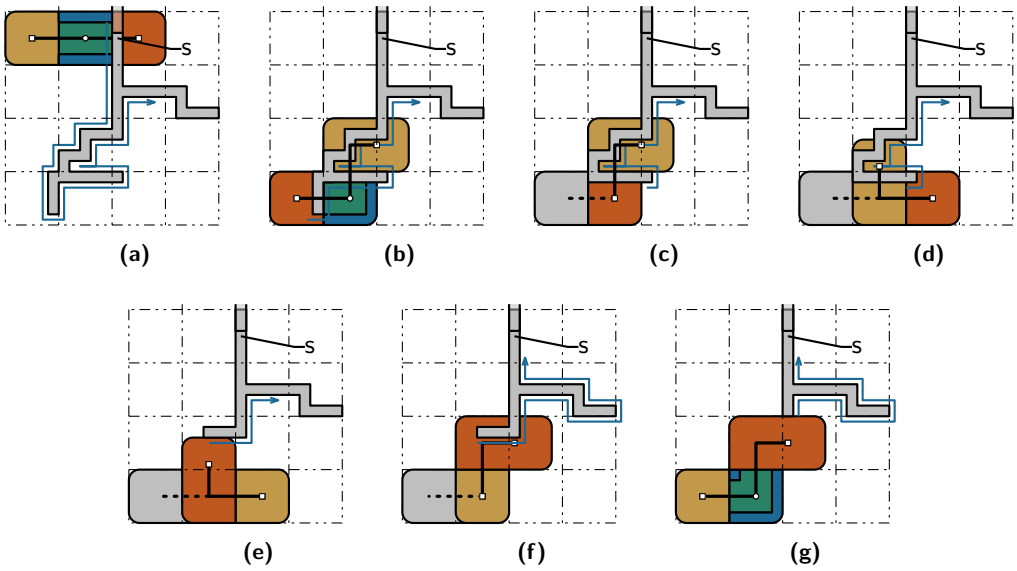


图12 深度优先搜索过程示例。在 (a)–(b) 中，蛇形结构移动到 T_s 的一个叶节点，并将其消耗，然后在 (c)–(d) 中分叉并推向其路径上的下一个元胞。在 (e)–(f) 中重复这一过程，之后在 (g) 中将分叉脊柱与之前的连通分量合并。

由“舍弃的”脊柱路径组成的树，这些路径将多个尾部连接到同一个头部；在任何时候，这棵树中都只有一条路径是活跃的蛇形结构。算法 1 给出了这一过程的伪代码描述，图12 则展示了包含若干次迭代的示例。

5.3 支架结构和压缩

由于篇幅所限，这一阶段的详细描述留在附录中。不过，图13提供了直观说明。将蛇形结构增长到适当大小后，我们可以移动到 C 的顶部，然后在该坐标轴 $z[C]$ 上构建 C 的投影，这就是我们的“支架结构”。我们将这一投影转换为一组大型蛇形结构，它们全部朝向 C 的底部移动，消耗 C 的模块。这会将 C 转换为一系列垂直蛇形结构，然后我们再将其转换为紧凑构型。

任意紧凑构型之间的高效重构在串行模型 [2, 16] 以及并行二维模型 [3, 12]中都已得到深入研究。来自 [3] 的技术，尤其是其中的技术，已经（尽管不一定能够直接实现）扩展到我们的三维设置中。为完整起见，我们在附录中展示了如何将元模块的紧凑构型高效地相互转换。

▶**引理15** （⋆）。 任意两个元模块的紧凑构型都可以通过完工时间为 $O(A + h)$ 的原位调度相互转换，其中 A 和 h 分别是两者沿某个坐标轴 $\lambda \in \{x, y, z\}$的最大投影面积和最大延展范围。

6 结论

我们给出了关于三维中滑动立方体并行移动的若干复杂度与算法结果。特别是，我们证明了该问题在完工时间或对称差大小方面不是固定参数可处理的，并且一般情况下是对数近似困难的。这是

16 并行滑动立方体

算法 1 T_{s} 的深度优先搜索遍历过程。

输入：一个脊柱为 $v_h, \ldots, v_t$ 的蛇形配置，以及序列 $\mathrm{m}1, \ldots, \mathrm{m}_\ell$。输出：一个完工时间为 $\mathrm{O}(\ell)$ 的调度，为该序列赋予蛇形所有权。 1：如果 $\mathrm{m}_1 \notin N_2^*[v_h, \ldots, v_t]$，则 2：$v_a \leftarrow$ 相邻元胞中的第一个主顶点，且该元胞与 m_1 相邻。 3：如果 v_a 是头顶点 $v_h \in P$，则 4：推拉，直到头顶点位于包含 m_1 的元胞中心。 5：否则 6：将后缀 $v_a, \ldots, v_t$ 分叉，并以 v_a 作为头顶点。 7：推拉，直到分叉后的头部位于包含 m_1 的元胞中心。 8：结束如果 9：结束如果 10：如果 m_1 是 T_{s} 的叶节点，则 11：根据引理 13 消耗 m_1 。 12：结束如果 13：如果可能，合并到一个规模更大的蛇形结构中。 14：使用得到的蛇形结构和 $\mathrm{m}2, \ldots, \mathrm{m}_\ell$ 递归。

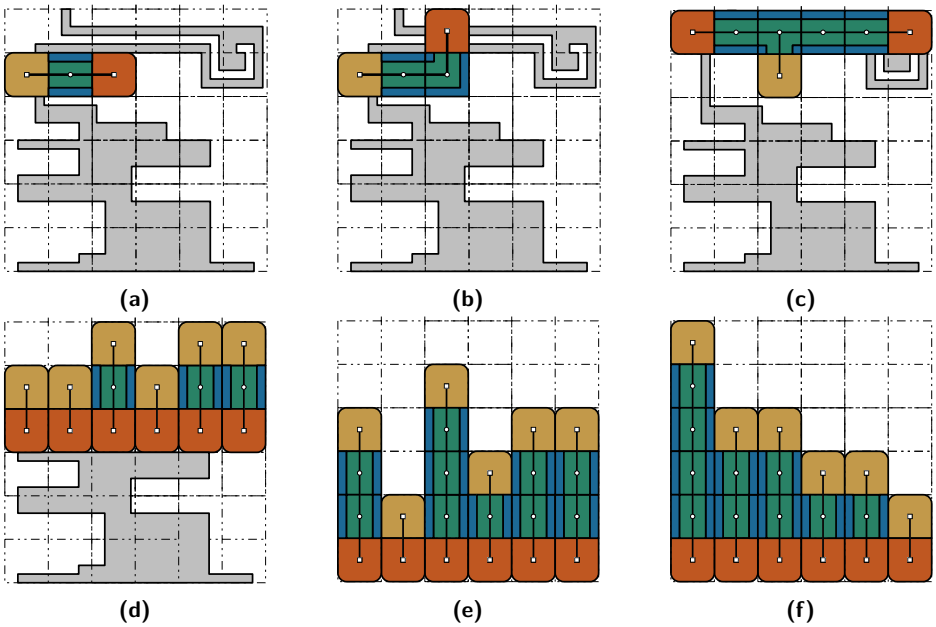


图13 支架与压缩概览。(a) 我们有一条大小为 $\mathcal{O}(A+h)$ 的蛇形结构。(b) 将蛇形结构移至 C 顶部。(c) 使用算法 1 填充 $z[C]$。(d-e) 使用蛇压缩构型。(f) 将生成的塔移向原点，从而紧凑化 C。

这是相关模型中的首个此类不可近似性结果，并强调了输入敏感型最坏情况最优算法的重要性；本文也提出了这样一种算法。

仍有一些尚未解决的问题。第4节的结果促使我们研究障碍物之间的滑动立方体；障碍物可以提供连通性，但无法移动。为了尽可能简化论述，引理6中的界很可能可以改进为 $\mathcal{O}(|S|^2)$。此外，我们用于得出复杂度结果的技术都无法推广到低维设置：二维与三维之间是否存在显著的跨越?

参考资料

1 Zachary Abel、Hugo A. Akitaya、Scott Duke Kominers、Matias Korman 和 Frederick Stock。《滑动立方体机器人的通用原位重构算法：以二次数量的移动完成重构》。载于计算几何研讨会（SoCG），第 1:1–1:14 页，2024年。doi:10.4230/LIPIcs.SoCG.2024.1。2 Hugo A. Akitaya、Erik D. Demaine、Matias Korman、Irina Kostitsyna、Irene Parada、Willem Sonke、Bettina Speckmann、Ryuhei Uehara 和 Jules Wulms。《正方形压缩：滑动方块的输入敏感原位重构》。载于斯堪的纳维亚算法理论研讨会暨研讨班（SWAT），第 4:1–4:19 页，2022年。doi:10.4230/LIPIcs.SWAT.2022.4。3 Hugo A. Akitaya、Sándor P. Fekete、Peter Kramer、Saba Molaei、Christian Rieck、Frederick Stock 和 Tobias Wallner。《并行滑动方块》。载于欧洲算法研讨会（ESA），第 28:1–28:17 页，2025年。 doi:10.4230/LIPIcs.ESA.2025.28。4 Hugo A. Akitaya、Matias Korman 和 Frederick Stock。《滑动立方体的输入敏感重构》。载于加拿大计算几何会议（CCCG），第 216–222 页，2025年。 arXiv:2507.04170。5 Giorgio Ausiello、Pierluigi Crescenzi、Giorgio Gambosi、Viggo Kann、Alberto Marchetti-Spaccamela 和 Marco Protasi。《复杂性与近似：组合优化问题及其可近似性》。施普林格，2012年。doi:10.1007/978-3-642-58412-1。6 Gruia Călinescu、Adrian Dumitrescu 和 János Pach。《图与网格中的重构》。SIAM离散数学杂志，22(1):124–138，2008年。 doi:10.1137/060652063。7 Mark de Berg 和 Amirali Khosravi。《平面线段的最优二元空间划分》。国际计算几何与应用杂志，22(3):187–206，2012年。 doi:10.1142/S0218195912500045。8 Erik D. Demaine、Sándor P. Fekete、Phillip Keldenich、Henk Meijer 和 Christian Scheffer。《协调运动规划：以有界伸展重构标记机器人群》。SIAM计算杂志，48(6):1727–1762，2019年。 doi:10.1137/18M1194341。9 Irit Dinur 和 David Steurer。《并行重复的解析方法》。载于计算理论研讨会（STOC），第 624–633 页，2014年。 doi:10.1145/2591796.2591884。10 Adrian Dumitrescu 和 János Pach。《推动正方形移动》。图与组合，22(1):37–50，2006年。 doi:10.1007/s00373-005-0640-1。11 Adrian Dumitrescu、Ichiro Suzuki 和 Masafumi Yamashita。《变形系统的运动规划：可行性、可判定性与分布式重构》。机器人学汇刊，20(3):409–418，2004年。doi:10.1109/TRA.2004.824936。12 Sándor P. Fekete、Phillip Keldenich、Ramin Kosfeld、Christian Rieck 和 Christian Scheffer。《具有有界伸展的连通协调运动规划》。自主智能体与多智能体系统，37(2):43，2023年。 doi:10.1007/S10458-023-09626-5。13 Sándor P. Fekete、Peter Kramer、Christian Rieck、Christian Scheffer 和 Arne Schmidt。《高效重构连通的标记机器人群》。自主智能体与多智能体系统，38(2):39，2024年。 doi:10.1007/S10458-024-09668-3。14 Robert Fitch、Zack J. Butler 和 Daniela L. Rus。《异构自重构机器人的重构规划》。载于国际智能机器人与系统会议（IROS），第 2460–2467 页，2003年。 doi:10.1109/IROS.2003.1249239。15 Ferran Hurtado、Enrique Molina、Suneeta Ramaswami 和 Vera Sacristán Adinolfi。《二维晶格模块化机器人系统的分布式重构》。《自主机器人》，38(4):383–413，2015年。doi:10.1007/S10514-015-9421-8。16 Irina Kostitsyna、Tim Ophelders、Irene Parada、Tom Peters、Willem Sonke 和 Bettina Speckmann。《滑动立方体的最优原位压缩》。载于斯堪的纳维亚算法理论研讨会暨研讨班（SWAT），第 31:1–31:14 页，2024年。doi:10.4230/LIPICS.SWAT.2024年.31。

17 Keith D. Kotay 与 Daniela L. Rus。自重构分子运动规划算法。载于国际智能机器人与系统会议（*IROS*），第 2184–2193 页，2000年。doi:10.1109/IROS.2000.895294。18 Othon Michail、George Skretas 与 Paul G. Spirakis。论可编程物质可行机制的变换能力。 计算机与系统科学杂志，102:18–39，2019年。 doi:10.1016/j.jcss.2018.12.001。19 Joel Moreno 与 Vera Sacristán。通过泛洪原地重构滑动方块。载于欧洲计算几何研讨会（*EuroCG*），第 32:1–32:7 页，2020年。URL：https://computational-geometry.org/abstracts/eurocg/2020.pdf。20 An Nguyen、Leonidas J. Guibas 与 Mark Yim。受控模块密度有助于重构规划。 算法与计算机器人学新方向，第 23–36 页，2001 年。21 Daniela Rus 与 Marsette Vona。晶体机器人：利用可压缩单元模块实现自重构。《自主机器人》，10:107–124，2001年。 doi:10.1109/MRA.2007.339623。22 Daniel Dominic Sleator、Robert Endre Tarjan 与 William P. Thurston。旋转距离、三角剖分与双曲几何。载于计算理论研讨会（*STOC*），第 122–135 页，1986年。 doi:10.1145/12130.12143。23 Matthijs Wolters。滑动方块的并行算法。硕士论文，乌得勒支大学，2024年。

A 第3节的缺失细节和省略的证明

A.1 NP完全性

▶**定理 1** （⋆）。 并行滑动立方体是NP完全的（完工时间1 且对称差大小1）。

证明。 我们从 平面单调 3-SAT归约，该问题询问给定的布尔公式是否可满足 [7]。每个子句至多包含 3 个文字，且要么全为正文字，要么全为负文字；子句-变量关联图必须存在一种平面嵌入，使变量映射到 x-坐标轴上，正（分别为负）子句映射到上（分别为下）半平面，并且边不与 x-坐标轴相交。令 $\deg(x_i)$ 表示包含变量 x_i的正文字的子句数，令 $\deg(x_i)$表示负文字对应的子句数。构造时，从 φ出发，我们布置多个由三种构件类型组成的实例，并加入一个表示对称差的模块，如图14所示。

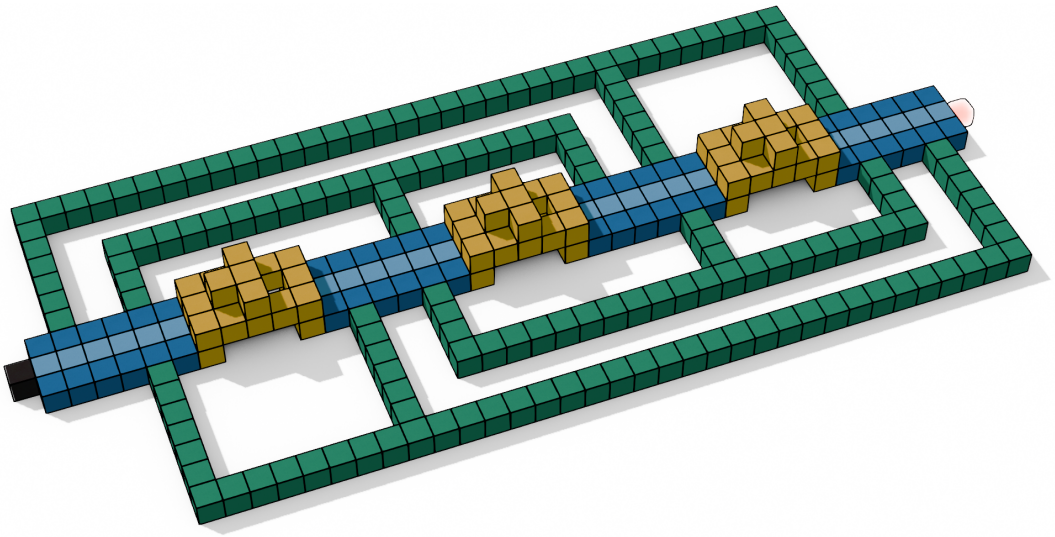
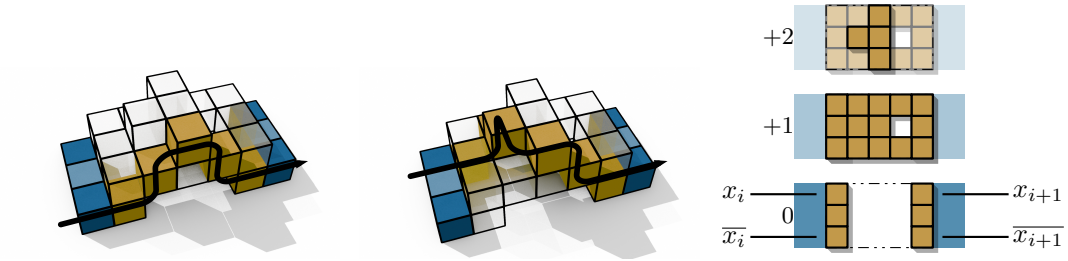


图14 我们针对 $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$的构造。颜色表示组件类型，变量构件从左到右按升序排列。

更准确地说，我们首先沿x坐标轴，在 xy-平面上放置 m 变量构件 （蓝色）。每个变量构件仅由一些模块组成，这些模块排列成一个矩形，其深度恰为 3、高度恰为 1，宽度为 $3{\cdot}\max\{\deg(x_i),\deg(x_i)\}$。我们利用模块沿 y坐标轴的坐标，区分变量构件中的正 模块和负模块；它们将在变换步骤中编码分配给每个变量的布尔值。变量构件沿 x坐标轴彼此相隔 5 个单位，为每对相邻变量构件之间的连接构件 （黄色）留出空间。连接构件占据一个宽度为 5、深度和高度均为 3 的包围盒；它是我们构造中唯一具有固有三维结构的构件。详细分解见图 3(b)。

此外，每个变量构件上还至少附有一个子句构件（绿色）。子句构件是“梳状结构”，与其包含的变量构件中的正模块或负模块相邻。正文字意味着，该构件有一个面相邻模块与相应变量构件中的正模块面相邻。最后，在初始配置中，我们沿着 x 坐标轴，在构造的一端放置一个差异模块（黑色／透明），并在目标配置中将其放置于另一端；请注意，这



(a) 路径位于连接器的一侧。 **(b)** 路径切换到另一侧。 **(c)** 连接构件的各层。

图15 连接构件及其相邻变量构件的示意图。

个模块是初始配置与目标配置之间唯一不同的连通分量。要通过单次变换重新配置所得实例，该模块实际上必须通过一连串不间断的移动推动其他模块，从初始位置“传送”到目标位置。

为了分析我们的构造，我们讨论这一系列移动所描述的路径的性质。由于仅存在于初始构型中的模块必须通过执行一次凸过渡立即移动，因此第一个变量的正模块或负模块也必须移动。此外，这一选择会在变量构件内部一致地传递：要么所有正模块并行移动，要么所有负模块并行移动。进一步说，这意味着，仅出现在目标配置中的位置，是由最后一个变量构件中的某个正模块或负模块通过执行凸过渡占据的。由于第一个变量将一个模块“推入”第一个连接器，而连接器在初始构型和目标配置之间保持不变，因此必定有一个模块离开连接器。因此，从黑色模块到白色位置之间必定存在连续且不间断的移动；这也意味着，每个变量构件要么移动其所有正模块，要么移动其所有负模块。

接下来说明连接构件中的可能移动方式。首先，假设第一个变量的正模块移动，并且第二个变量的正模块也打算移动。这会使第一个变量中与连接模块相邻的正模块通过一次滑动移动推入连接器。对称地，离开连接器的模块也通过同一侧的一次滑动移动离开。随后，被挤开的模块必须进行一次凸过渡，而移入空出位置的模块也必须进行一次凸过渡。能够实现这种行为的唯一合法移动序列，是在同一侧进行两次凸过渡，如图15(a)中的路径所示。现在，假设第二个变量的负模块打算移动。在这种情况下，连接器内部的移动方式有所不同，如图15(b)中的路径所示。具体而言，最顶层中的模块可以通过凸过渡从一侧旋转到另一侧。因此，连接器的整体形状得以保持，但其中一个模块移动到了先前由第二个变量的负模块占据的位置。

由于模型中的骨干约束，这意味着在变换过程中，所有子句都会与其所含文字不匹配的变量构件断开连接，并且任何可行路径都与 φ 的一个满足赋值一一对应。具体而言，将一个变量设为真会触发其负变量模块的移动。这会使任何连接到该变量负侧的子句与该变量断开，因此这些子句必须通过其他设为假的变量连接。◀

我们现在论证如下：

▶**推论 2** (⋆)。 除非 P = NP，否则即使对称差的大小为常数，也无法在多项式时间内将并行滑动立方体近似到优于2 的因子。

证明。 只需证明，按照定理 1 中归约所构造的实例，可以在不满足底层布尔公式的情况下，通过两次变换求解。我们的策略如下。我们忽略所有正子句，并仅沿变量构件的负侧移动模块。

在第一步中，我们将负模块序列向右移动，并移入相邻的连接构件，使每个子句构件仍与其所连接的、索引最小的变量保持连接。在第二步中，我们重复这一模式，同时继续维持索引最小的连接。这样，在两次变换内，我们就可以将差异模块从构型的一端传送到另一端。◀

A.2 对数 APX-困难性

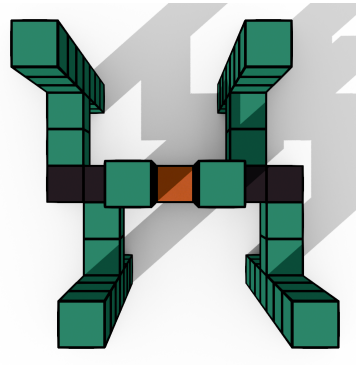
现在，我们将介绍之前概述中省略的 log-APX-困难归约的详细内容。其中包括对构造及其构件的全面描述，随后是对整个归约的细致分析。为清晰起见，我们将归约分为两部分。首先，我们分析一个简化的变体，称为不可移动变体，该变体更易于分析。在此变体中，我们假设尖刺模块（图中描绘为绿色立方体）是不可移动的，也就是说，它们无法移动。然后我们将说明，一旦对构件进行轻微调整，在原始变体中所采用的假设仍然成立。

我们的归约来自对数近似困难问题集合覆盖 [9]。一个实例$\Sigma_{n,\ k}$由自然数 1，……，n 以及一个由$\{s_1$，……，$s_k\}$ 中的子集组成的集合构成，其中$\{1$，……，$n\}$ 满足 $\bigcup_{i=1}^{k} s_i = \{1$，……，$n\}$。任务是找到一个基数最小的集合 $S^\star$，使得 $\bigcup_{s_i \in S^\star} s_i = \{1$，……，$n\}$。我们用 ℓ 表示子集 S' 的大小，用 $\ell^\star$ 表示最优解 $S^\star$的大小。

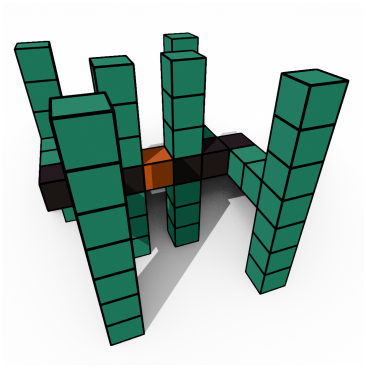
我们首先详细描述所有构件。在图中，绿色立方体表示尖刺。尽管图中这些尖刺向上和向下延伸三层，但实际上它们由更多立方体组成；具体数量将在后文说明。在整个证明过程中，我们用 C_I表示滑动序列的初始构型，用 C_F表示最终构型。

覆盖构件。对于每个 $j = 1$，……，$n \in \mathbb{N}$，我们引入一个覆盖构件，如图4中的紫色方块所示，详见图16。在覆盖构件中，不可移动模块的路径被一个“可移动”模块m_j打断，该模块在图16中以橙色突出显示。在 C_I中，橙色模块位于0层，而在 C_F 中，它被抬升至 $+1$层，恰好位于其原始位置正上方。不可移动模块路径上紧邻橙色模块的两个模块都带有向上和向下的尖刺；除了用于固定立方体之外，这些是不可移动变体中唯一发挥作用的尖刺。在不可移动变体中，我们假设这些尖刺是不可移动模块。

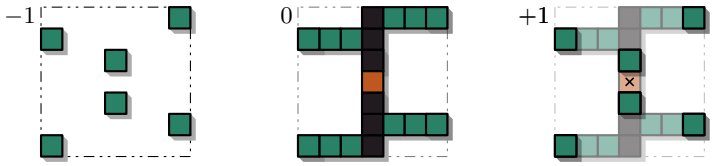
层构件。 为了将图4扩展到第三维，我们引入 层构件，如图所示，它们是标有黑色箭头的蓝色方框；另见图17(a)。层构件表示路径向上弯折，升高三个层，随后恢复原来的方向。在图4中，箭头表示路径从较低部分过渡到较高部分。



(a) 覆盖构件的俯视图。



(b) 覆盖构件的侧视图。



(c) 覆盖构件的三层：一层位于可移动模块和不可移动路径上方，另一层位于下方。

图16 覆盖构件。橙色立方体表示可移动模块，其起始位置位于第0层，目标位置位于第 $+1$层。深色立方体表示不可移动模块。绿色立方体对应尖刺。

选择器构件。 该选择器构件以蓝色方块表示。此构件用于表示某个数 j属于 1、.....、 n构成的集合 s_i。在选择器构件中，一条不可移动模块路径与第二条路径相互垂直。第一条路径的末端模块与第二条路径相距1 。每条路径都连接到与数 j对应的覆盖构件，但连接方向不同。移动一个移动机器人以连接这两条路径，即可建立骨干约束，从而使覆盖构件中的橙色模块能够移动。

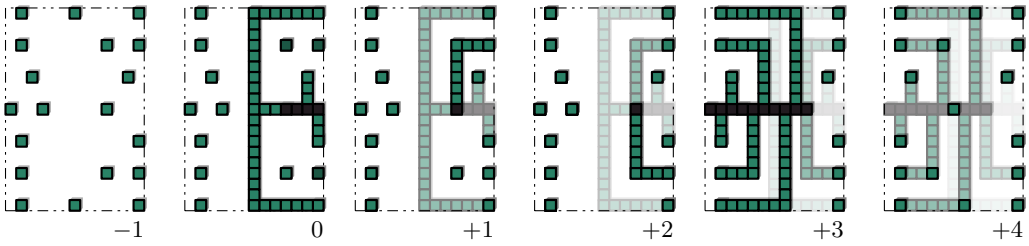
数字构件。 为了编码数字 j，我们沿着一条水平路径构造链式构件（如图4中标记为 1 的灰色矩形所示）。其右侧包含一个覆盖构件。带橙色斜线的方框用作占位符，表示实际路径远比图中所示的长；为便于阅读，图中对其进行了压缩。对于每个包含 j 的集合，一条通向集合构件（标记为 的灰色矩形）的附加垂直路径会通过一个 ***T***形交叉口从主水平路径分支出去。

组合所有数字1、.....、 n：每个数字1、.....、 n分别由一条专用的水平路径表示。这些路径在右侧连接到一把梳状结构。最大的数字置于最前方，占据所有数字构件中最低的层。在任意两个相邻的数字构件之间，我们插入一个层构件，使得 $j-1$ 的构件位于 j的构件上方三层；参见图4最右侧垂直路径旁的浅蓝色方格。这确保了通向选择器构件的路径，对于每个数字构件都起始于不同的垂直位置。



(a) 一条向上攀升三个单位的路径.....

(b) .....上面附有许多尖刺。



(c) 路径构件的三层。

图17 层构件将路径构件跨越垂直距离（以3的倍数计）连接起来。

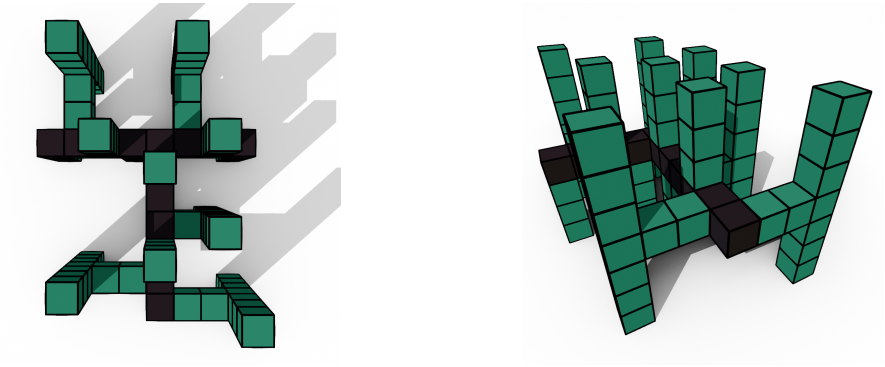
集合构件：为了表示集合 $s_i \in S$，我们引入集合构件（标有一个写着 s_i的灰色矩形）。集合构件由一条路径组成，该路径在图4中从最底部的行垂直延伸至灰色矩形。其上端设有与数字构件相连的选择器；当且仅当 $j \in s_i$时，数字 j 对应的选择器才存在。它们按如下顺序排列：路径上位置越高的选择器对应的数字越大。

组合集合构件： 图4中最底部的水平路径与集合构件共同构成一个梳状结构。集合构件按照其索引指定的顺序从左向右连接。在 C_I中，可移动模块 m 位于构件与 s_1 相连处的拐角，而在 C_F 中，它位于构件与 s_k 路径相连处的T形交叉口。

综合起来：由集合构件形成的梳状结构与由数字构件形成的梳状结构相交于一个点（如图4右下方所示）。两条路径交汇处的立方体距离所有集合构件和数字构件都相当远。橙色斜线填充矩形表示这一较大的间距。

接下来，我们将讨论图4所示构件之间的各种空间间隔。

- 两个相邻选择器构件之间的距离设为 $B = 30$。因此，数字构件与选择器构件之间的垂直距离也为 B。对于连接到数字 k 的路径， $90°$ 转弯（即路径从水平转为垂直的位置）发生在水平距离为 $B(k + 1 - j)$ 之后。这确保转弯后相邻路径之间的垂直距离保持为 B。
- 为避免干扰，两个集合构件之间的垂直距离设定为：前一个构件中所有选择器的 $90°$ 个转弯都位于更靠左的位置。该距离设为 $C = 2 \cdot B \cdot k$。



(a) 一个 T 形交叉构件。 **(b)** 更清晰的视图。

(c) T 形交叉构件的三层。

图 18 T 形构件将三个在同一点相交的路径构件连接起来。

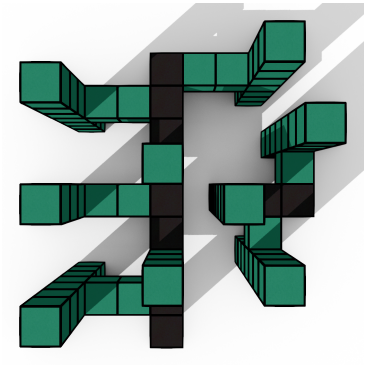
黑色斜线填充矩形表示的路径长度为 $A = k \cdot C + 2k(n-1)B + n$。橙色斜线填充矩形表示的路径长度为 $100 \cdot k \cdot A$。请注意，该长度也表示尖刺构件超出路径上下方的距离。

我们给出若干结果，以便对不可移动变体中将 C_I 重新配置为 C_F 所需的步数这一问题的计算复杂度得出结论。在继续之前，我们想先说明几点。(1) C_I的对偶图是树，其顶点对应模块，且当且仅当两个对应模块共享一个二维面时，这两个顶点之间才有边。(2) m 是 C_I中唯一能够执行移动的移动模块。显然，m$_j$ 无法滑动，因为这样做会将树分成两部分，从而违反骨干约束。若要让一个 m$_j$ 构件移动，我们可以将 m 移至一个选择器构件，并用它填补该分隔器的两条路径之间的间隙。这样，我们就构造出了一个同时包含 m 和 m$_j$的环，因此 m$_j$ 可以滑动至其相应的目标位置，而不违反骨干约束。

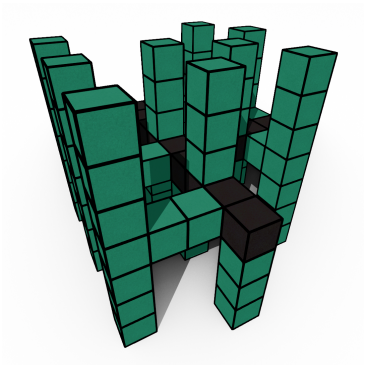
现在，我们为不可移动变体中将C_I 重新配置为 C_F 所需的移动次数建立一个简单的上界。这样我们就能排除最短序列中的某些事件，因为执行这些事件就会超出该上界。

▶**引理 16**。 在不可移动变体中，存在一个将 C_I 变为C_F 的并行滑动序列，其长度为$2 \cdot k \cdot A + k \cdot (C+6) + 2k(n-1)(B+6) + n$。

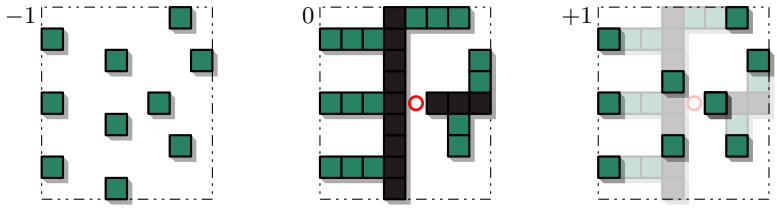
证明。 可移动模块 m 最初放置在左下角，其中 s_1 的集合构件连接到位于底部的水平路径。我们执行一系列顺序移动，将 m 移动到数字 j_1 在集合构件内的第一个选择器



(a) 一个选择器构件。



(b) 更直观的视图。



(c) 选择器构件的三层。

图 19 选择器构件使覆盖构件能够执行其操作。

s_1。接下来，m 移动到选择器构件中两条路径之间的间隙。这便形成了一个环，该环也包含模块 m_{j_1}。因此，我们可以执行一次滑动，将 m_{j_1} 从其初始位置移至最终位置，因为 m 提供了骨干。将 m_{j_1} 滑动到最终位置后，我们把 m 从间隙中移出，并将其移到某个编号为 j_2 的下一个选择器构件。我们再次将 m 移入选择器构件两条路径之间的间隙，将 m_{j_2} 移至最终位置，然后把 m 从间隙中移出。我们对 s_1 的集合构件中的所有选择器构件重复这一过程。处理完所有选择器构件后，我们将 m 移回 s_1 的集合构件与水平路径连接的位置。

接下来，我们将 m 移动到集合构件 s_2 与水平路径相连的点。集合构件 s_2 的处理方式与 s_1 相同，唯一的区别是，我们会跳过与 s_1 中已处理数字相连的选择器构件。对每个 $s \in S$ 对应的所有集合构件重复这一过程。最终，m 到达其最终位置；由于 $\bigcup_{i=1}^{k} s_i = \{1, \ldots, n\}$，覆盖构件中的每个 m_j 都已移动到其最终位置。

在单个集合构件中上下移动 m 需要 $2 \cdot A$ 次移动，所有集合构件总共需要 $2 \cdot k \cdot A$ 次移动。此外，在单个集合构件中访问所有选择器构件最多需要 $2(n-1)(B+6)$ 次移动，所有集合构件总计为 $2k(n-1)(B+6)$ 次。“+6”表示朝上的尖刺；经过构件时，需要额外滑动一次才能越过它：从下往上经过时一次，从上往下经过时一次。还需要额外滑动两次，才能将 m 插入选择器构件两条路径之间的间隙，并将其移出。从一个集合构件的连接点移动到另一个集合构件需要 $(C+6)$ 次移动，总共为 $(k-1)(C+6)$ 次。这里，“+6”同样表示在转弯构件和 T 形构件中朝向上方尖刺移动所需的额外移动。最后，将每个 m_j 从初始位置移至目标位置，每个模块需要滑动一次，因此总共需要 n 次移动。◀

26 并行滑动立方体

根据引理 16，我们可以推导出 C_I 与 C_F之间任意最短滑动序列都具有以下关键性质。

►**引理17**。 在不可移动变体的任意最短滑动序列中，任意 m_j *for any* $j = 1, \ldots, n$ 都恰好参与一次滑动操作，即从其初始位置滑动至目标位置。

证明。 我们首先注意到，“其目标位置”一词确实有明确定义：事实上，对于不同的 $j_1 \neq j_2$，若要将一个移动模块从 j_1 的初始位置滑动到 j_2 的目标位置，就必然需要使它穿过两个橙色斜线填充矩形，如图4所示。这至少需要 $200 \cdot k \cdot A$ 次滑动，超过了引理 16 给出的平凡上界。

考虑一个从 C_I 到 C_F 的任意滑动序列，其长度严格短于平凡上界。在此序列中，移动模块 m_j 可能参与多次滑动操作。然而，只有当 m 位于与 j的数字构件相连的选择器构件的间隙中时，它才会参与滑动。对于每个包含数字 j的集合 i ，令 sel(i, j) 表示相应的选择器构件。令 sel$(\cdot\ j)$ 表示与 j 的数字构件相连、且 m 曾跨接其两条路径之间间隙的所有选择器构件的集合。对于每个 $j = 1, \ldots, n$，我们恰好选择一个选择器构件 sel(i, j)，并将其纳入集合 sel*。现在，我们按如下方式构造一个滑动序列：初始时，m 位于其起始位置。我们沿着由所有集合构件组成的梳状结构的水平路径，将 m 从其初始位置滑动到最终位置。每当 m 到达一个包含选择器构件 sel$(i, j) \in$ sel*的集合构件时，我们就将 m 向上滑入该集合构件。在其中，我们让 m 进入属于该集合构件的每个选择器构件，以连接两条路径。这样，我们便能用每个橙色模块一次滑动，将每个 m_j 移动到其目标位置。

所得滑动序列上升的集合构件不多于原序列，使用的选择器构件不多于原序列，且对橙色可移动模块执行的滑动也不多于原序列。此外，只要原序列中任一橙色模块参与了不止一次滑动，所得序列对橙色模块执行的滑动就严格更少。因此，我们构造的滑动序列长度至多等于原序列的长度，并且只要原序列使任一橙色模块滑动超过一次，其长度就严格更小。

我们得出结论：在任一最短滑动序列中，任一橙色可移动模块至多参与一次滑动。◄

►**命题 18.** 在不可移动变体中，存在一个从C_I 到 C_F、长度为 $2 \cdot \ell \cdot A + (k-1) \cdot (C+6) + 2k(n-1)(B+6) + n$ 的并行滑动序列，当且仅当 $\Sigma_{n,k}$具有大小为 ℓ 的覆盖。

证明。 假设 $\Sigma_{n,k}$ 具有大小为 ℓ 的覆盖 S' 。我们执行如下移动序列。自由模块从左向右移动：每当它到达与 S'中某个集合相对应的集合构件时，便上升该构件，并在每一个连接到尚未被变换的覆盖构件的选择器构件中执行一次移动。

从左向右移动最多需要 $k \cdot (C+6)$ 次移动。随后转换选择器构件，每个元素需要移动一次，总计 n 次移动。上升一个集合构件需要滑动 A 次，下降则还需要额外滑动 A 次。模块对 ℓ 个集合构件重复这一过程。因此，集合构件总共需要 $2 \cdot \ell \cdot A$ 次滑动。在一个集合构件内部，模块仍须遍历所有选择器构件。每个集合构件最多可能需要 $2(n-1)(B+6)$ 次滑动，因此总计最多需要 $2k(n-1)(B+6)$ 次滑动。

现在假设 C_I 与 C_F 之间的滑动距离至多为 $2 \cdot \ell \cdot A + k \cdot (C+6) + 2k(n-1)(B+6) + n$。为了在覆盖构件内完成所有必要的滑动，可移动

模块 m 必须到达包含相应数字的选择器构件。为此，它必须沿集合构件上下移动。根据滑动序列的总长度，它最多只能上行和下行

$$\begin{aligned}
&\leq \left\lfloor \frac{2 \cdot \ell \cdot A + k \cdot (C+6) + 2k(n-1)(B+6) + n}{2 \cdot A} \right\rfloor \\
&= \left\lfloor \frac{2 \cdot \ell \cdot A}{2 \cdot A} + \frac{k \cdot (C+6) + 2k(n-1)(B+6) + n}{2 \cdot A} \right\rfloor \\
&= \left\lfloor \frac{2 \cdot \ell \cdot A}{2 \cdot A} + \frac{k \cdot (C+6) + 2k(n-1)(B+6) + n}{2 \cdot (k \cdot (C+6) + 2k(n-1)(B+6) + n)} \right\rfloor = \ell
\end{aligned}$$

许多集合构件。令 $S' \subset S$ 表示其集合构件已被模块上下移动过的集合。由于所有 m_j 都能够移动到各自的目标位置， S' 构成一个大小至多为 ℓ 的覆盖。◀

▶ 注 19。改写命题 18 中的估算可知，给定滑动距离 d，我们可以通过计算 $\ell = \lfloor 2^{d} \cdot A \rfloor$ 来确定覆盖 ℓ 的大小，或者等价地，计算 $\ell = 2^{d-R} \cdot A$，其中某个正数 $R < ((k-1) \cdot (C+6) + 2k(n-1)(B+6) + n)$。特别地，$\ell^* = 2^{d^*-R^*} \cdot A$，其中 $R^* < ((k-1) \cdot (C+6) + 2k(n-1)(B+6) + n)$。

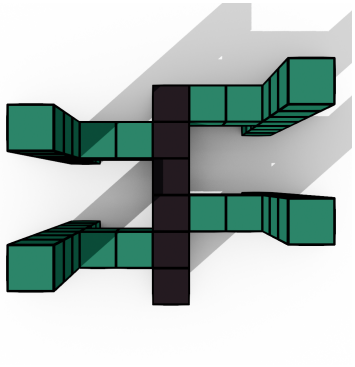
显然，不可移动变体中的任何有效滑动序列对于并行滑动立方体也同样有效。接下来需要证明的是，在没有不可移动模块的情况下，允许多个立方体并行滑动并不会带来任何改进。为此，我们在许多不可移动模块上连接一个尖刺，使得移除某个不可移动模块后，其尖刺会与构型的其余部分断开，从而违反骨干约束。图 20 展示了最简单的尖刺放置方式：沿着一条直线路径放置尖刺。在这里，尖刺按三种方向交替排列：一个向左延伸，下一个向右延伸，再下一个向下延伸。不过，这种交替模式无法在整个构造中统一应用。如图 21 所示，在转弯处，我们暂时让尖刺交替指向上方、下方和转弯外侧，以保持连通性。

图 16–22 分别展示了在交叉构件、T形交叉构件、选择器、层构件和覆盖构件上的其他尖刺连接方式；其中，交叉构件具体涉及一条水平路径和一条垂直路径，两者的高度差至少为 3。

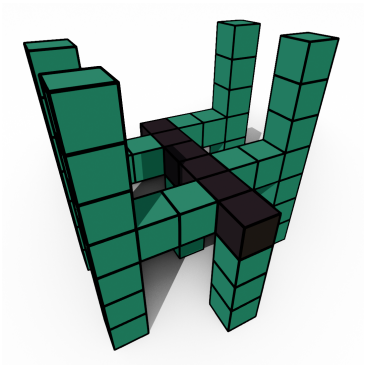
在图中，所有尖刺都用长度为 3、指向上方和下方的直线表示。一旦它们延伸到图 4 所示实例之外，尖刺就会向外扩展，以占据更多空间。具体而言，我们要求每个尖刺的端点与任何不属于同一尖刺的模块之间至少相距 $100 \cdot k \cdot A$。请注意，图 4 中的实例可以用一个关于原始集合覆盖问题规模的多项式 p 来描述，该多项式由 A、B 和 C 的取值决定。不可移动模块至多有 p 个，每个模块都由其自身的尖刺固定。为确保每个尖刺的端点与其他尖刺上的所有模块保持所需距离，如图 23 所示，大立方体的一侧面积必须为 $(100 \cdot k \cdot A)^2 \cdot p$。这一面积要求仍然是关于原始集合覆盖实例规模的多项式。

▶引理 *20. </style}*不失一般性，只需考虑并行滑动立方体的有效滑动序列，同时这些序列也是不可移动变体中的有效滑动序列。

证明。我们证明，对于任意长度短于由引理 16 得到的滑动序列，都存在另一个长度不超过它的滑动序列，其中指定为不可移动的模块都不会移动。在滑动序列开始时，只有m以及位于尖刺端点的模块可以自由移动。



(a) 一条路径构件。



(b) 更清晰的视图。



(c) 路径构件的三层。

图 20 我们归约的基础组件是路径构件。

位于尖刺端点的模块与任何不属于尖刺的模块之间的距离为 $100 \cdot k \cdot A$。因此，在最短的滑动序列中，它们不可能被移动到 C_I 的任何构件中。此外，它们无法与自身尖刺之外的模块构成环，也无法释放构件附近的任何模块。因此，任何会移动尖刺末端模块的非平凡滑动序列，都可以替换为一个等效序列：执行相同的滑动，但不移动这些尖刺末端模块。新的滑动序列仍会将C_I 变换为 C_F，且长度不超过原序列。

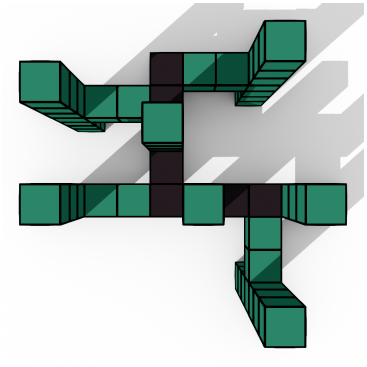
因此，要移走在不可移动变体中被视为不可移动的模块，可移动模块必须充当骨干。我们声称这是不可能的。原因如下：所有不可移动模块 M 和尖刺模块都属于以下三类之一：

(i) 移除 M 后，C_I 会分裂成 C_I 个连通分量，而且任何额外模块最多只能同时连接其中两个连通分量。**(ii)** 移除 M 后，C_I 会分裂成两个连通分量，重新连接它们的唯一方法是在 M原来的位置放置一个模块。**(iii)** 移除 M 后，C_I 会分裂成两个连通分量，且 M 是连接到一个交叉构件上的尖刺的一部分。由于所有交叉构件与任意模块 $\mathrm{m} \cdot k \cdot A$ 之间的距离至少为 m 和 $\mathrm{m}j$（对任意 $j = 1, \ldots, n$）都至少相距 100，因此，不存在比平凡滑动序列更短的滑动序列，能够让一个移动体移动足够远，从而满足骨干约束并释放一个模块。

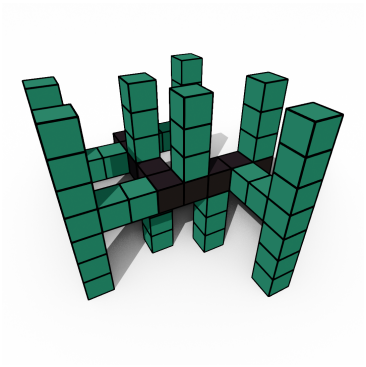
在所有情况下，我们都可以得出结论：对于任意最短滑动序列，都存在一个长度相同的序列，其中没有不可移动模块或尖刺模块发生移动。◀

为了最终证明并行滑动立方体的对数 APX-困难性，我们进行一种所谓的 AP-归约。

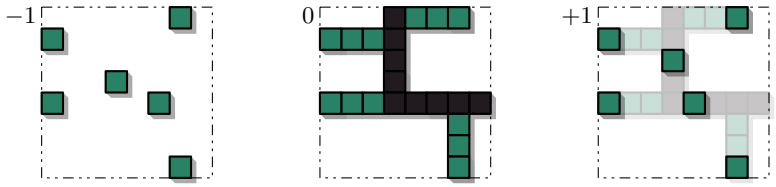
▶定义 *21*（[5, 第 *257*–258] 页）。设 P_1 和 P_2 是两个 *NP*优化问题。如果存在两个函数 f 和 g 以及一个常数 $\alpha \geq 1$，使得对于 P_1 的任意实例 X 和任意有理数 $r > 1$，$f(X, r)$ 都是 P_2的一个实例，则称 P_1 可 *AP* 归约到 P_2。



(a) 一个转弯构件。



(b) 更直观的视图。



(c) 转弯构件的三层。

图 21 转弯构件将路径构件正交连接起来

- 对于 P_1 的任意实例 X 以及任意有理数 $r > 1$，如果 X存在可行解，那么 $f(X, r)$存在可行解。对于 P_1 的任意实例 X、任意有理数 $r > 1$以及 $f(X, r)$的任意可行解 Y，X的可行解为 $g(X, Y, r)$。f 和 g 均可在多项式时间内计算。对于 P_1 的任意 X、任意有理数 $r > 1$以及 $f(X, r)$的任意可行解 Y，如果 Y的性能比至多为 r，则 $g(X, Y, r)$的性能比至多为 $1 + \alpha(r - 1)$。

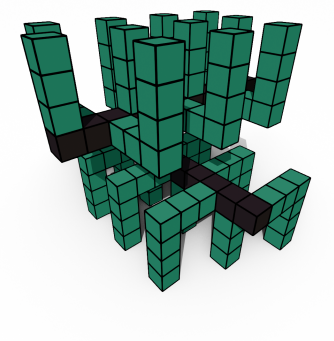
在我们的情形中，f 对应于从集合覆盖到并行滑动立方体的归约，而 g 是一个操作：按照命题 18 的证明中所述，我们通过滑动序列获得一个集合覆盖。我们要求 f 和 g 是多项式时间可计算的。对于常规的 AP-归约，较弱的要求就足够了：对于任意固定的 r，只需 f和 g 是多项式时间可计算的。这是证明某个问题为 APX-困难的标准技术。相比之下，要证明对数 APX-困难性，则需要更强的条件：f 和 g 的运行时间必须独立于 r。我们进一步选择 $\alpha = 4$。

▶**定理 4** ($\star$)。并行滑动立方体和顺序滑动立方体都是对数 APX-困难；因此，除非P =NP，否则无法以优于 $\Theta(\log(n))$的因子进行近似。

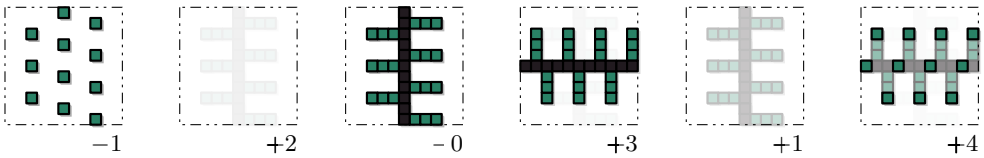
证明。假设我们可以将最优滑动序列的长度近似到至多 r 倍的因子。那么我们就能计算出一个长度至多为 $d < r \cdot d^*$ 的滑动序列



(a) 垂直距离为 3 的路径交叉……



(b) ……附有大量尖刺。



(c) 六层由两个交叉路径构件组成。

图 22 对于交叉构件，路径会暂时放弃完整的垂直尖刺，改为向上或向下延伸。这会削弱尖刺约束：交叉点必须远离空闲模块。

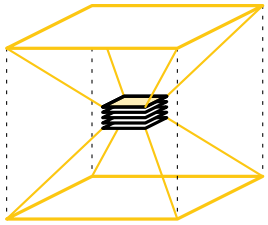


图 23 如何将尖刺投影到大立方体边界上的示意图。

多项式时间。根据备注 19，这使我们能够得到一个大小为

$$\ell = \frac{d-R}{2\cdot A} < \frac{r\cdot d^*-R}{2\cdot A} = \frac{r\cdot d^* - r\cdot R^* + r\cdot R^* - R}{2\cdot A}$$
$$= \frac{r\cdot d^* - r\cdot R^*}{2\cdot A} + \frac{r\cdot R^* - R}{2\cdot A} < r\cdot \ell^* + \frac{r\cdot R^*}{2\cdot A} < r\cdot \ell^* + \frac{r}{2}$$

现在，我们得到了一个基于 ℓ^* 和 r 对 ℓ 的估计。下一步是将其改写为定义 21 所指定的形式，即 $\ell \le \ell^*(1+\alpha(r-1))$。

为此，我们区分两种情况。首先，假设 $(r-1) \ge \frac{1}{2\ell^*+1}$。我们作如下估计

$$r\cdot\ell^* + \frac{r}{2} = \ell^* + \frac{1}{2} + (r-1)\ell^* + \frac{(r-1)}{2} + (3\cdot\ell^* - 3\cdot\ell^*)$$
$$= \ell^* + 4(r-1)\ell^* + \frac{1}{2} - (r-1)(3\cdot\ell^* - \frac{1}{2}) \le \ell^*(1+\alpha(r-1))$$

对于第二种情况，假设 $(r-1) < \frac{1}{2\ell^*+1}$。我们按以下方式进行估计。

$$r \cdot \ell^* + \frac{r}{2} = \ell^* + \frac{1}{2} + (r-1)\ell^* + \frac{(r-1)}{2} < \ell^* + (r-1)(\ell^* + \frac{1}{2}) + \frac{1}{2} < \ell^* + 1$$

在第二种情况下，该估计使我们能够恢复最优解；而在第一种情况下，不等式如预期般成立，并满足 AP-归约的要求。至此，我们的证明完成。◀

B 第4节省略的细节与缺失的证明

▶**引理5** ($\star$)。 任意连通构型的外部扩展面中都存在一个空闲模块。

证明。 令$\mathcal{F}(C)$ 表示构型 C的外部扩展面所包含的模块面集合。我们对 $|\mathcal{F}(C)|$进行归纳。

基本情形：考虑一个构型 C，其中 $|\mathrm{F}(C)| \leq 3$。这只有在$C = \mathrm{F}(C)$时才可能。如果 $|\mathrm{F}(C)| = 1$，那么 C 是一个模块，而且显然是空闲模块。否则， C 有两个空闲模块，因为每个连通构型都有两个空闲模块（引理 6 [21]）。由于 $C = \mathrm{F}(C)$，因此 $\mathrm{F}(C)$中有两个空闲模块。

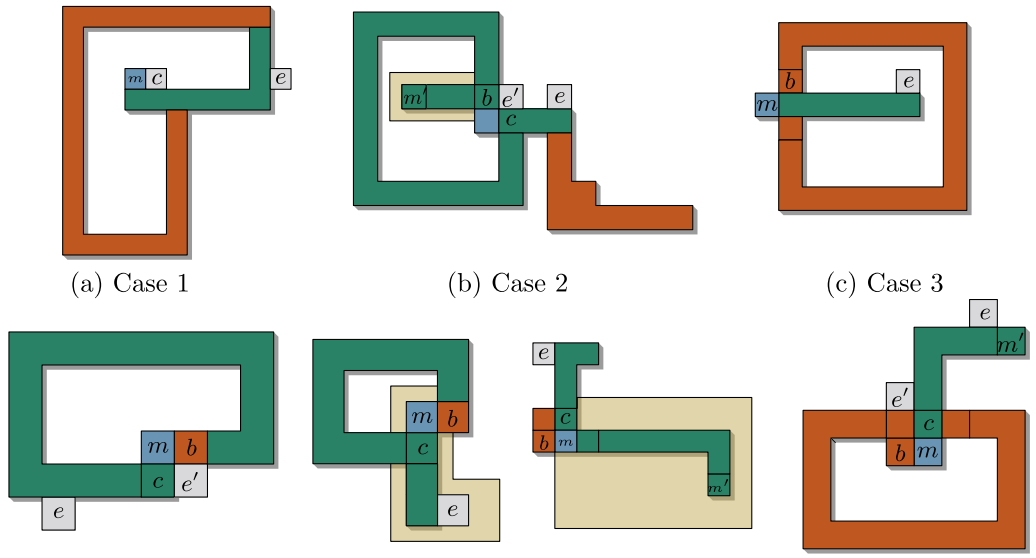
现在假设，对于任意外部扩展面大小为 $|\mathcal{F}(C)| < k$ 的 C，$\mathcal{F}(C)$ 中都存在两个空闲模块（除非 $|C| = 1$）。考虑一个满足 $|\mathcal{F}(C)| = k$ 的构型 C。我们确定两个极端模块：最高、最靠左、最顶层的模块 m_{max}，以及最低、最靠右、最靠下的模块 m_{min}。由于它们处于极端位置，这两个模块都属于外部扩展面。

接下来我们检查以下情况：如果 m_{min} 或 m_{max} 是自由的，引理便成立。如果二者都不自由，则考虑 $C \setminus \mathrm{m}_{max}$，它至多有三个连通分量，因为 m_{max}最多只有三个邻居 a、b、c。

我们从这些连通分量中任选一个，不失一般性地，设其包含 a,K_a。如果 m_{max} 是一个割模块，那么 b 或 c 至少有一个不是 K_a 的元素。同样，不失一般性地，设 b $\notin K_a$，或者 $K_b \cap K_a = \emptyset$。我们声称 $\mathcal{F}(K_a) < \mathcal{F}(C)$。注意，此时 a 和 b 在 $C \setminus \mathrm{m}_{max}$ 中构成一条夹边。a 和 b 的公共邻域中有两个单元格，其中一个原先被 m_{max} 占据，另一个我们称为 z。后者 z 必须为空，否则 a 和 b 会在 $C \setminus m_{max}$ 中连通。如果 $\mathcal{F}(K_a)$ 实际上大于 $\mathcal{F}(C)$，那么与 z 相邻的 a 的面，记作 f_z，必定属于 $\mathcal{F}(K_a)$，但不属于 $\mathcal{F}(C)$。

为了得出矛盾，假设情况确实如此（$f_z \in \mathcal{F}(K_a)$，但 $f_z \notin \mathcal{F}(C)$）。显然，b 和 c 也与 f_z 相邻。考虑 b,c 与 a,c 之间的公共边。如果 $f_z \notin \mathcal{F}(C)$，则这些边都不能是 C 中的夹边。然而，只有当 a 与 c 以及 b 与 c 都有一个公共邻居时，这才可能。但如果这些公共邻居存在，那么 a、b 和 c 都会彼此相连，并且 $K_a \cup K_b \neq \emptyset$，从而产生矛盾。因此，$|\mathcal{F}(K_a)| < |\mathcal{F}(C)|$，并且我们的归纳假设适用。$\mathcal{F}(K_a)$ 至少有两个空闲模块。因此其中至少一个不是 a，从而该模块在 C中是自由的。我们也可以对 K_b 进行归纳，以得到另一个自由模块，从而满足我们的归纳假设。◀

▶ **引理6** ($\star$). 给定一个构型 C，一个连通子集 $S \subset C$，满足 $|S| \geq 2$，且$C \setminus S$ 连通；一个与 S相邻的空单元格 e；以及一个自由模块 $m \in S$，使得 e 和 m 位于 $S \setminus \{m\}$的同一个扩展面上，并且 $(S \setminus \{m\}) \cup \{e\}$ 和 $(C \setminus \{m\}) \cup \{e\}$ 均为连通集合。存在一种调度，其完工时间为 $2^{\mathcal{O}(|S|)}$，可将 m 传送至 e。



(a) Case 1 (b) Case 2 (c) Case 3

(d) Case 3a (e) Case 3b (f) Case 3c (g) Case 3d

图24 引理6证明中的情况。

证明。我们对 $|S|^2 + \delta_F(\mathrm{m}\ e)$ 使用归纳法，其中 $\delta_F(\mathrm{m}\ e)$ 表示 m 与 e 沿着 $S \setminus \{\mathrm{m}\}$中包含二者的扩展面 F 之间的距离。也就是说，在由 F诱导的模块面的邻接图中，$\delta(\mathrm{m}\ e)$ 是 m 在 F中的模块面与 e 在 F中的模块面之间最短路径的长度。注意，$|S|$ 是 $\delta_F(\mathrm{m}\ e)$的上界。基本情形是 m 与 e 滑动相邻，且 m 可以移动到 e，此时结论成立。注意，m 与 e 相邻且 m 可以通过 e进行凸过渡的情形不可能发生，因为那会使 $(C \setminus \{\mathrm{m}\}) \cup \{e\}$ 不连通。

现在我们对归纳步骤进行分类讨论。注意，如果存在一系列移动，将 m 带到 e S 的表面上，那么该断言显然成立。因此，以下我们假设不存在这样的调度，即所有此类路径都被阻断。请参见图24。

情况1。 m 与一个空单元格滑动相邻 c，其中 $\delta_F(c,\ e) < \delta_F(\mathrm{m}\ e)$。那么我们将 m 移动到 c，归纳假设成立。

情况 2。从 m 到 e 的路径被模块 b $\in S$阻断。那么，b 在$S \setminus \{\mathrm{m}\}$中有一条与另一个模块 c $\in S$ 相连的夹边，并且 m 与二者都相邻。令 e' 为同时与 b 和 c 相邻的另一个单元格。注意，$e' \notin C$，否则 m 到 e 就不会位于同一个扩展面内。如果 b 是自由的，则将其滑动到 e'，再将 m 滑动到 b 的原位置。然后，通过交换 m 和 b 的标签，归纳假设成立。否则，令 S' 为 b 的一个不包含 c 或 m 的割分量。根据引理5，我们可以在 S' 中找到一个自由模块 m′ ≠b。随后，我们可以在 S' 中使用归纳假设，将 m′ 传送到 e'，然后再将 m 传送到 m′的原位置，从而有效地将 m 传送到 e'。于是，归纳假设成立，因为 $\delta_F(e', e) < \delta_F(\mathrm{m}\ e)$。

情况3。 从 m 到 e 的路径被模块 b $\in C$阻挡。令 c $\in S$ 为 S中 m 被 b阻挡时与之相邻的模块。要么m 已经与 b 相邻，要么可以滑动到与 b相邻的位置。因此，我们可以假设 $(C \setminus S) \cup \{\mathrm{m}\}$ 是连通的。我们分成几个子情形。

情况3a。c空闲。于是，如果c和b在 $C \setminus \{m\}$中构成一条夹边，令 e' 为同时与c和b相邻的空单元格。注意，根据b的定义， $\delta_F(e', e) < \delta_F(m\ e)$。然后，我们可以将c移至 e'，并将m移至c原来的位置，从而有效地将m传送到 e'。接着，归纳假设成立。如果c和b没有构成夹边，则有一个更简单的操作。我们可以利用归纳假设将c移至 e，然后将m移至c原来的位置。

情况3b。 c 是 $S \setminus \{m\}$ 中的一个割模块，且连通分量 S' 属于 $S \setminus \{c,m\}$ 与 e 不相邻，但与 $C \setminus S$相邻。那么， $C \cup S'$ 是一个连通构型。接着，我们可以对 $S \setminus S'$ 应用归纳假设，将 m 传送到 e 上 $S \setminus S'$的一个扩展面。

情况3c。c 是 $S \setminus \{m\}$ 中的一个割模块，且 $S \setminus \{c, m\}$ 中所有不与 e 相邻的连通分量也不与 $C \setminus S$ 相邻。于是，m 要么与某个 S' 相邻，要么可以移动到与某个 S' 相邻的位置。我们应用归纳假设，将一个自由模块 $m' \in S'$（由引理5可知其存在）传送到 e（位于 $S \setminus \{m\}$ 中）。然后再次应用归纳假设，将 m 传送到扩展面上 m' 的原始位置，该扩展面属于 $S' \cup \{m\}$。

情况 3d。c 在 $S \setminus \{m\}$ 中不是割模块，这意味着删除它会将两个连通分量分开： $C' = (C \setminus S) \cup \{m\}$ 和 $S' = S \setminus \{c, m\}$。接着，存在一个同时与 C' 和 S' 相邻的空位置e'，否则就会存在一条从 m 到 e 的直接路径。这个单元格必须更靠近 e，因为它与 S' 相邻。如果 $|S| \geq 3$，我们可以应用归纳假设，将自由模块 $m' \in S'$ 传送到 $S \setminus \{m\}$ 上的 e'。我们可以再次应用归纳假设，将 m 传送到 $S \setminus \{m'\}$ 扩展面上 m 的原始位置 $'$。这实际上将 m 传送到了 e'；由于 $\delta_F(e', e) < \delta_F(m\ e)$，归纳步骤填充了 e。最后，如果 $|S| = 3$，将 m' 移到 e' 会使 S 断开。通过将 m' 移到 e'、将 c 移到 m' 的原始位置，再将 m 移到 c 的原始位置，可以得到与上述相同的结果。

我们可以直接将上述归纳法转化为递归算法。为此，令$T(|S|, \delta_F(m\ e))$表示递归算法的运行时间。最坏情况递推式出现在情况 3d 中，此时：

$$T(|S|, \delta_F(m, e)) \leq 2 \cdot T(|S| - 1, |S| - 1) + T(|S|, \delta_F(m, e) - 1).$$

然而，在这种情况下， $T(|S|, \delta_F(m\ e) - 1)$项实际上关于 $|S|$ 是线性的，因为我们可以保证从 m' 到 e'存在一条直接路径。因此，执行的移动次数为 $2^{\mathcal{O}(|S|)}$。◀

C 第 5 节中省略的细节和缺失的证明

C.1 寻找 T_s

▶ **引理 22.** 给定一个常数 $t \in [n]$， T 包含一棵以模块 p 为根的子树 T_p，使得 $t \leq |T_p| \leq 5t$。

证明。 结论由以下事实得出： T 的最大度为 6。如果 $n/5 \leq t \leq n$，T_r 满足结论，其中 r 是 T 的根。否则，r 有一个子节点 p 满足 $|T_p| > t$，因为 r 最多有 5 个子节点。然后我们可以对 T_p 递归。◀

我们利用引理 22 得到一棵大小为 T_s $\Theta(A + h)$ 的树。

C.2 省略的证明

▶ **引理8** (⋆)。 给定一个大小为 $|T_m| \geq 2625$ 且为 $|T_m| \in O(1)$ 的 T 的子树 T_m，存在一个 $O(1)$ 调度，使得生成的构型中包含 m 的元胞已满，并且 $T \setminus T_m$ 中的每个单元格都保持为满。

证明。 设 $(\mathrm{m}, \mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3, \mathrm{m}_4)$ 是 T_{m} 中的一条路径（由 T_{m}的大小可知该路径存在）。我们首先声称，可以构造一个以m_1, m_2, 或 m_3为中心、半径为 2 的球。注意，根据构造， T 具有以下性质：

(M) 设 T_{m} 是一个以 m 为根节点的子树。如果 $\mathrm{p} \in T_{\mathrm{m}}$ 与同一元胞中的某个模块 q 相邻，那么 q 要么是 $\mathrm{p} = \mathrm{m}$ 的父节点，要么 $\mathrm{q} \in T_{\mathrm{m}}$。

▷ 命题 23. 在 $\mathcal{O}(1)$ 的完工时间内，存在一个调度，可以填满一个以 m_1,m_2 或 m_3为中心、半径为 2 的 L_1 球。

证明。 设 $\mathcal{B}_1$ 和 $\mathcal{B}_2$ 分别表示对应于以 L_1 为中心、半径为 2 的球的单元格集合，球心分别为 m_1 和 m_2, 。设 S_{m} 为最初位于 T_{m}中的模块集合。我们进行如下分类讨论。

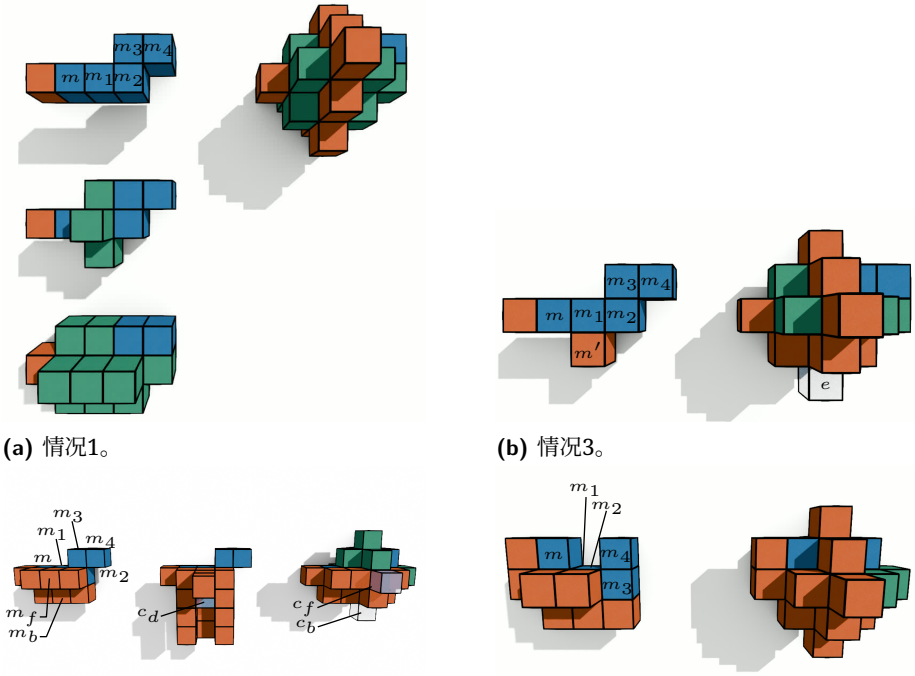
情况1。 假设 m_1 不在其元胞的边界上。那么，根据 $N(\mathrm{m}_1)$的定义，其中的每个单元格都必须为空，或属于 T_{m}。如果它们为空，我们可以应用引理6，使用 $S = S_{\mathrm{m}}$ 以及该连通分量之外的一个自由模块。该连通分量属于 $T_{\mathrm{m}} \cap \mathcal{B}_1$ 且包含m；由于 T_{m}的大小，这样的自由模块必定存在。因此，我们可以假设 $N(\mathrm{m}_1)$ 已被 S_{m}中的模块填满。同样，如果$\mathcal{B}_1$ 中仍有空单元格，我们可以应用引理6将其填满，因为每个这样的空单元格都与 S_{m}相邻。

情况 2。 如果 m_1 在其元胞之外的所有邻居要么为空，要么属于 S_{m}，则与第一种情况相同的论证同样适用，并且我们可以反复应用引理6，直到 $\mathcal{B}_1$ 被填满。

情况3。 不失一般性，假设 m_1 位于其元胞的底面，且其底部邻居 m' 是唯一一个不属于 S_{m} 的模块邻居。那么，与情况1类似，应用引理6可以填满 B_1, 中的所有位置，除了 m'下方的单元格 e 。如果 e 已经填满，我们就完成了。否则，注意到 m' 是自由的：它原来的每个不属于 S_{m} 的邻居现在都与 S_{m}中的一个模块相邻，而 S_{m} 通过 m 与其余构型相连；m 不可能是 m'的邻居。还要注意，$\{\mathrm{m}_1, \mathrm{m}'\}$是自由的，因为 B_1 除 e外已被填满。因此，我们可以使用$S = \{\mathrm{m}_1, \mathrm{m}'\}$应用引理6，将 m_1 传送到 e。这样，m_1 原来的单元格就变为空单元格，而 $S_{\mathrm{m}} \setminus \{\mathrm{m}_1\}$仍保持连通。由于该单元格与 $S_{\mathrm{m}} \setminus \{\mathrm{m}_1\}$相邻，我们可以应用引理6将其填满。

情况4。无损一般性地，假设 m_1 位于其元胞的底部前侧边缘，并且其底部（($(-z)$方向）和前方（($(-y)$方向）的邻居 m_b和 m_f 是仅有的不属于 S_{m} 的邻居。注意，情况1–3也适用于 m_2。因此，我们还可以假设 m_2 位于其元胞的底部前侧边缘；否则，我们将对 m_2应用前述某种情况。我们还假设 m_2 恰好有两个不属于 S_{m} 的邻居。（如果它只有一个，则适用情况3。）使用与情况1相同的论证，我们可以填满 B_1 中的所有位置，但以下位置除外：位于 m_f 前方的单元格 c_f （$\mathrm{m}_1 + (0, -2, 0)$）、位于 m_f下方的单元格 c_d （$\mathrm{m}_1 + (0, -1, -1)$），以及位于 m_b 下方的单元格 c_b （$\mathrm{m}_1 + (0, 0, -2)$）。我们分情况讨论。

情况4a. 如果 c_d 是满的，而 c_b 是空的，则适用与情况3相同的论证，并且根据引理6，我们可以将m_1 传送到 c_b 。对称地，当 c_d 是满的，而c_f 是空的时，结论同样成立。



(a) 情况1。 **(b)** 情况3。

(c) 情况4。 **(d)** 情况5。

图25 命题23的证明。模块 m_1, m_2, m_3 和 m_4 以蓝色显示，保证属于 S_{m} 的模块以绿色显示，而不保证属于 S_{m} 的模块以红色显示。

情况4b. 现在，假设 c_d 是空的。如果m_b 空闲，则集合{m_1,m_b}空闲，因此我们可以将 m_1 传送到 c_d ，并像情况3中那样填满m_1 原来的单元格。m$_f$的情形也相同。如果m_b 不空闲，那么 c_b 必定是满的，因为m_b 的其他每个邻居现在都与 S_{m}相邻。因此，删除 m_b 会使 c_b 处的模块与m_b的其他邻居断开连接。于是，单元格 $c_b + (1, 0, 0)$和 $c_b + (-1, 0, 0)$ 也必定为空。这意味着 m_2 下方的模块 m_{2b} 是空闲的（ c_b +(1, 0, 0) 或 c_b +(−1, 0, 0)之一位于m_{2b} 下方，且它的其他所有邻居都与B_1中的其他满模块相邻）。如果m_{2b} 能够向 c_d进行凸过渡，我们就可以将m_2 向下移动，并根据引理6填补m_2 留下的空位置。否则，m_{2b} 与 c_d 的公共邻居是满的。在这种情况下，我们可以对m_2 应用情况4的第一个子情形，因为对应的$c_d = \mathrm{m}_2 + (0, -1, -1)$是满的。

情况5. 假设 m_1 有3个不属于 S_{m}的邻居。不失一般性，m_1 是其元胞左下角的前侧顶点。那么，m_2 和 m_3 都位于同一元胞的一条边上，因此它们各自至多有两个不属于 S_{m}的邻居。于是，我们可以重复情况4，将 m_1 和m_2 分别替换为 m_2 和m_3。 ◁

一旦得到一个半径为2的球，就可以利用以下命题将其扩展。设 $\mathcal{B}$ 是以模块 m^* 为中心、由命题23构造得到的满单元的最大 L_1 球。令 S' 为 $T_{\mathrm{m}} \setminus \mathcal{B}$ 中在构造 $\mathcal{B}$后仍保留原位的模块集合。

▷ 命题 24。设$\mathcal{B}$ 是以模块 m 为中心的最大满单元 L_1 球。如果 L_1- 半径 radius($\mathcal{B}$)大于或等于 2，则存在一个完工时间为$\mathcal{O}$(radius($\mathcal{B}$))的调度，可将 m 传送到$\mathcal{B}$相邻的一个空位置。

证明。 我们对 $\mathcal{B}$的半径使用归纳法证明该命题。归纳步骤中，假设$r =$半径$(\mathcal{B}) > 2$。

归纳步骤。 假设对于半径为 $r-1$ 的任意球，该断言都成立。我们称 $\mathcal{B}$ 中的一个单元格为极端单元格，如果它是沿某个坐标轴方向最大的或最小的 6 个单元格之一。我们希望填充一个 e 与 $\mathcal{B}$ 的边界相邻的空位置。我们区分两种情况：

情况1。 存在一个空位置 e，它与模块 u $\in\mathcal{B}$ 相邻，且该模块不是极端模块。由于 $r > 2$，并且 u 不是极端模块，u 在 $\mathcal{B}$ 内具有足够的连通性：它不是割点，因为所有不在 $\mathcal{B}$ 中的邻居也与 $\mathcal{B}$ 中的其他模块相邻。我们将 u 移动到 e。这会在 u 的原位置产生一个空位，记为 p_{vac}，它位于半径为 $r-1$ 的球的边界上。根据归纳假设，我们可以将中心模块 m^* 传送到 p_{vac}，以填充该位置。

情况 2。 与 $\mathcal{B}$ 相邻的所有空单元格都只与 $\mathcal{B}$ 的极端模块相邻。不失一般性，假设 e 是位于 u （$\mathcal{B}$ 的最顶层模块）上方的空单元格。根据情况 $N^*(\mathsf{u}) \setminus \{e\}$ 的定义，该位置是满的，从而确保 u 是自由的：所有不在 $\mathcal{B}$ 中的邻居也与 $\mathcal{B}$ 中的其他模块相邻。我们将 u 向上朝 e 移动。根据局部支撑的不同，这有两种方式：

C情况 2a（直接滑动）。 如果 e 与另一个被占据的单元格相邻，u 便直接滑入 e。然后我们应用归纳假设，填补 u 原始位置处的空位。

情况 2b（凸过渡）。 如果 e 没有其他被占据的邻居来支撑滑动，u 便执行凸过渡，到达临时位置 e'，该位置与 e相邻。这腾出了u的原始位置 p_{vac}。然后我们应用归纳假设，将 m^* 传送到p_{vac}。一旦 p_{vac} 被重新填满，它便为 u 提供必要支撑，使其从 e' 滑动到最终目的地 e。

基本情形。 基本情形发生在 $\mathcal{B}$ 的半径为 2 时。

情况1。 假设与 $\mathcal{B}$ 相邻的仅有空位置在某一基本方向上处于极端位置。不失一般性，设 $e = \mathsf{m}^* + (0,0,3)$。则集合 $\{\mathsf{m}^*, \mathsf{m}^* + (0,0,1), \mathsf{m}^* + (0,0,2)\}$ 是空闲的，并且它们都可以在一次并行移动中向上移动。

情况 2。 假设 $e = \mathsf{m}^* + (1,\ 0,\ 2)$ 为空。将将下面的模块 $e\,(\mathsf{m}^* + (1,\ 0,\ 1))$ 上移至 e。如果模块 $\mathsf{m}^* + (0,\ 0,\ 1)$ 空闲，则将其右移至 $\mathsf{m}^* + (1,\ 0,\ 1)$，然后将 m^* 上移至 $\mathsf{m}^* + (0,\ 0,\ 1)$。否则，模块 $\mathsf{m}^* + (0,\ 0,\ 1)$ 不空闲，因此，删除它会使 $\mathsf{m}^* + (0,\ 0,\ 2)$ 与 B 的其余部分断开。因此，$\mathsf{m}^* + (0,\ 0,\ 2)$ 的所有水平相邻单元格（具有相同的 z 坐标）都是空的，否则它们会将其与 $\mathsf{m}^* + (1,\ 0,\ 1) \in \mathcal{B}$连接起来。我们将 $\mathsf{m}^* + (1,\ 0,\ 1)$ 移至 e，并进一步分解为子情形。

情况 2a。假设 $\mathsf{m}^* + (1, -1, 2)$ 为空。于是，我们可以将 e 中的模块进行凸过渡，使其移动到 $\mathsf{m}^* + (0, -1, 2)$。这将 $\mathsf{m}^* + (0, 0, 2)$ 与 $\mathsf{m}^* + (0, -1, 1) \in$ B 连接起来，从而腾出 $\mathsf{m}^* + (0, 0, 1)$。接下来，我们可以像之前一样，将 $\mathsf{m}^* + (0, 0, 1)$ 向右移动，并将 m^* 向上移动。由于我们在 B 相邻的位置填入了一个空位置，因此操作完成。

情况 2b。假设 $\mathsf{m}^* + (1, -1, 2)$ 和 $\mathsf{m}^* + (1, -1, 1)$ 都已填满。那么，$\mathsf{m}^* + (0, 0, 1)$ 是空的（参见路径 $(\mathsf{m}^* + (0, 0, 2)\ e, \mathsf{m}^* + (1, -1, 2), \mathsf{m}^* + (1, -1, 1), \mathsf{m}^* + (0, -1, 1))$），我们可以按照标准情况 2，将 $\mathsf{m}^* + (0, 0, 1)$ 向右移动，并将 m^* 向上移动。

情况 2c。假设 $\mathsf{m}^* + (1, -1, 2)$ 已占满，而 $\mathsf{m}^* + (1, -1, 1)$ 为空。然后，我们将 $\mathsf{m}^* + (1, -1, 0)$ 中的模块向上移至 $\mathsf{m}^* + (1, -1, 1)$。接着，我们可以应用情况 2b。当 $\mathsf{m}^* + (1, 0, 1)$ 占满后，$\mathsf{m}^* + (1, -1, 1)$ 对连通性而言便不再是必需的，我们可以将其中的模块向下移回其原始位置 $\mathsf{m}^* + (1, -1, 0)$。

情况3. 假设$\mathsf{m}^* + (1, -1, 1)$ 为空，且$\mathsf{m}^* + (0, 0, 2)$ 的所有水平相邻单元格均为满。然后，我们可以将模块 $\mathsf{m}^* + (1, 0, 1)$ 移动到 $\mathsf{m}^* + (1, -1, 1)$。

由于 $\mathrm{m}^* + (0,-1,2)$ 为满，$\mathrm{m}^* + (0,0,1)$ 为空闲，我们可以按情况 2 进行，将 $\mathrm{m}^* + (0,0,1)$ 向右移动，并将 m^* 向上移动。◁

我们使用命题 24 填充$\mathcal{B}$ 旁边的一个空位置，并清空$\mathcal{B}$的中心。如果 S'在$\mathcal{B}$ 旁边有一个自由模块，那么我们就可以反向使用命题 24，填充$\mathcal{B}$的中心。我们从 S' 中删除这个模块，因为它现在已成为球$\mathcal{B}$的一部分。如果 S' 在$\mathcal{B}$旁边没有自由模块，那么必定存在一个同时与 S' 和$\mathcal{B}$相邻的空位置。这是因为根据 T 和 S'的定义， S' 中的模块只与 S' 中的其他模块以及$\mathcal{B}$中的模块相邻，而命题 24 保证 S' 与$\mathcal{B}$相邻。然后，我们可以根据引理6，将 S' 中的一个自由模块传送到$\mathcal{B}$ 旁边的一个空位置。重复这一过程，我们可以扩展$\mathcal{B}$ ，直到它包含整个元胞。◀

C.3 支架与压缩的细节

在将蛇形结构生长到适当大小后，我们构建一个“支架结构”，这是一种临时子结构，使我们能够并行执行许多蛇形操作，以“压缩”该构型，并最终重新配置为紧凑构型。一旦蛇形结构达到 $\mathcal{O}(A+h)$ 的大小，我们便停止生长并开始构建支架结构，它将是 $z[C]$ 的缩放版本。然后，我们利用该支架结构将 C “压缩”为紧凑构型。

我们使用推拉操作，将蛇形结构的头部移动到高度等于 C 的最大模块的位置。假设头部与 B_1 的一个 $5\times5\times5$ 元胞对齐。然后，我们在 $z[C]$ 的缩放单元格上计算一棵生成树 T ，并以包含蛇形结构头部的元胞为根节点。我们使用算法 1 遍历 T。请注意，由于蛇形结构的起始大小为 $\mathcal{O}(A+H)$，蛇形结构的初始元胞数量多于 T 的元胞。因此，在此次遍历结束时，我们的蛇形结构可以占据 T 的每一个元胞。我们在 T 的每个元胞处分叉，分裂成许多蛇。

我们现在得到了支架结构：一个由蛇构成的连通分量，其中每条蛇的头部都处于同一高度；我们将这些头部的整体记为 ß。我们可以通过推操作并行地将每条蛇向下移动。请注意，对于单条蛇而言，在推操作的每个阶段，大部分外壳单元都是静止的。因此，ß 中的每条蛇都可以并行执行推操作，并且它们的外壳将保持连通性。当一条蛇遇到模块 $\mathrm{m} \in C \setminus$ß 时，该蛇消耗$\mathrm{m}$。

当 ß 到达 C 的底部时，我们终止该过程。此时， C 是一组按 $z[C_1]$ 的缩放投影排列的垂直蛇形结构。我们现在重新排列这些蛇形结构，以形成紧凑构型。首先，我们“压缩”每条蛇，通过将尾部的模块“传送”到任意空的脊柱位置来填补空的脊柱单元。这样便得到一个由实心垂直塔组成的构型，即对于每个模块 $\mathrm{m} \in C$，m 正下方的任意单元格均为满。以下步骤将重新配置 C ，使得 m 左侧的每个位置均为满：

1. 取每个 y 偶数模块 y 坐标。
   a. 在这个集合中，将每个 z 坐标为偶数的模块向左滑动，除非它会移到 x 坐标为负数的位置，或者它左侧的相邻模块不会移动。（没有左侧相邻模块的模块必须先向上或向下进行凸过渡，然后滑入相应位置）b. 现在，将剩余的 y 坐标为偶数且 z 为奇数的模块向左滑动，除非它会移到 x 坐标为负数的位置，或者它左侧的相邻模块不会移动。2. 现在，按照相同的步骤，将每个 y 坐标为奇数的模块向左滑动：先处理 z 坐标为偶数的模块，再处理 z坐标为奇数的模块。3. 重复以上步骤，直到 C 相对于 y坐标轴单调

交替处理偶数和奇数的 z 坐标，可以确保连通性得以保持。当所有 y 和 z 坐标为偶数的模块移动时，每个 y坐标为奇数的模块都通过那些 y 坐标为偶数且 z坐标为奇数的模块保持连通。对于 y/z 坐标的其他三种奇偶组合，同样的论证也成立。经过这一过程后，对于任意模块 m，它正下方的每个单元格仍然都是满的。此外，m 正左方的每个位置也都满足这一点，否则我们就不会结束这一过程。这种关系具有传递性，因此对于每个这样的 m，所有 z 或 x 坐标更小的单元格都是满的。

如果重复这一过程，将任何对“y 坐标”的引用替换为“x 坐标”，并将“向左移动”替换为“向下移动”。 C 现在是紧凑构型，因此仍然成立：对于每个 m，所有 z 或 x 坐标较小的单元格都是满的。然而，通过对称论证，所有 y 较小的位置也都是满的。这就是紧凑构型的定义。

▶**引理 9**。 如果一个构型包含一个半径为12的完全占据的 L_1 球，则它包含一个自由蛇形构型，其脊柱路径长度为11 ，并且有198个模块。

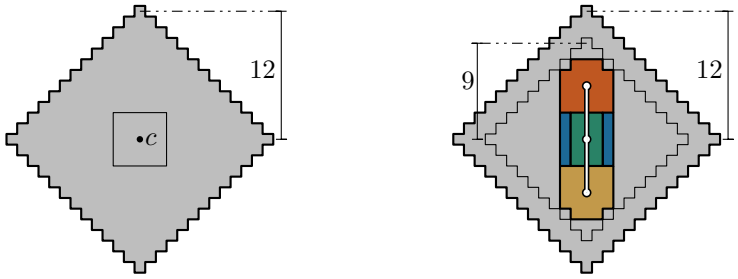


图 26 L_1 球的一个截面及其包含的蛇形结构。

证明。设 C 是一个构型，其中包含一个半径为 12、中心位于 $c \in \mathbb{Z}^3$ 的 L_1 球。我们用 $P = (c - (0,0,5) \ldots, c + (0,0,5))$ 表示一条长度为十一的脊柱路径，并证明 $S(P)$ (i) 诱导出一个有效蛇形构型 $S(P) \cap C$，且 (ii) 对应于 C 的一个自由子构型。回忆一下，根据定义， $S(P) \subset N_4^*(P)$。此外，脊柱路径的每个顶点都在中心 c 的五个单位范围内，即 $P \subset N_5^*(c)$。根据三角不等式可知， $S(P) \subset N_9^*(c)$，因此蛇形子配置完全包含在球内。由于 L_1 球还剩下三层完整的外层， $S(P)$ 是自由的。◀

▶**引理 10** ($\star$)。 推操作可以通过完工时间为 $O(1)$ 的调度来完成。

证明。 考虑一个蛇形子配置 $S(P) \subseteq C$ ，令 $v_h \in \mathbb{Z}^3$ 为定义蛇形结构头部的脊柱端点 P 。不失一般性，假设$v_h = (0,0,0)$ ，且推操作的方向朝上。令位于 $z = 2$ 的蛇头的九个外壳单元为 旧前沿 F_c，它们的上方邻格为 新前沿 。

首先说明，我们可以通过清空头部位于 $z(v_h)$ 的八个已占用内部单元以及 v_h 上方的模块，逐步填满新前沿。

▷ 命题25。如果 F_c 中的模块未空闲，则 F_n 中面相邻的单元格已被占用。

证明。 设 m $\in F_c$；我们根据其开放 L_1邻域 $N(c)$进行论证。请注意，旧前沿中的每个模块都位于头部模块面邻接关系中的一个 4环上，也就是说，该面邻接关系是二顶点连通的。特别地，这意味着，如果 m 不是自由的，那么必定存在一个不属于头部的相邻模块。

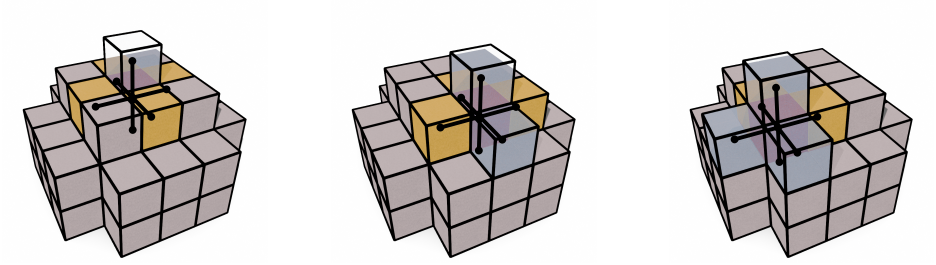


图 27 当前前沿中的单元格要么空闲，要么与新前沿中的模块相邻：对于 $\mathrm{m} \in F_c$（红色），我们将 $N(\mathrm{m})$ 分成蛇形结构内的单元格（黄色）和蛇形结构外的单元格（蓝色、透明）。

对于 $x(\mathrm{m}) = y(\mathrm{m}) = 0$，结论显然成立：$N(\mathrm{m})$ 中唯一不在头部的单元格是其上方邻格，而这也是 m 可能不自由的唯一原因，见图 27(a)。现在假设 m 是旧前沿的一个外层模块，例如 $x(\mathrm{m}) = 1$ 以及 $y(\mathrm{m}) = 0$；可能的情形见图 27(b) 和图 27(c)。如果 m 不自由，且其上方邻格未被占据，那么必定存在一个位于其东侧邻格中的模块，使其成为割模块。然而，该单元格中的模块必定要与 m 一同位于构型的一个公共环上，因为它与头部中的两个模块相邻，而头部本身是二顶点连通的。◁

为了填充新前沿单元格 $n \in F_n$，我们现在取其下方最多四个单位高度列中的模块，一直向下到 $z(v_h)$，并执行向上并行滑动，见图 28(a)。请注意，如果新前沿中没有任何边相邻单元格被占据，最顶层模块可能无法执行向上滑动。在这种情况下，该模块会改为执行凸过渡，然后滑动到新前沿中所需的单元格，如图 28(b) 所示。

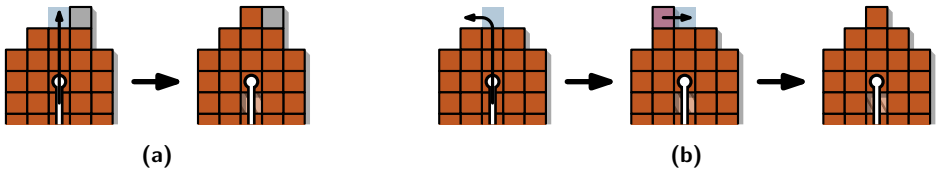
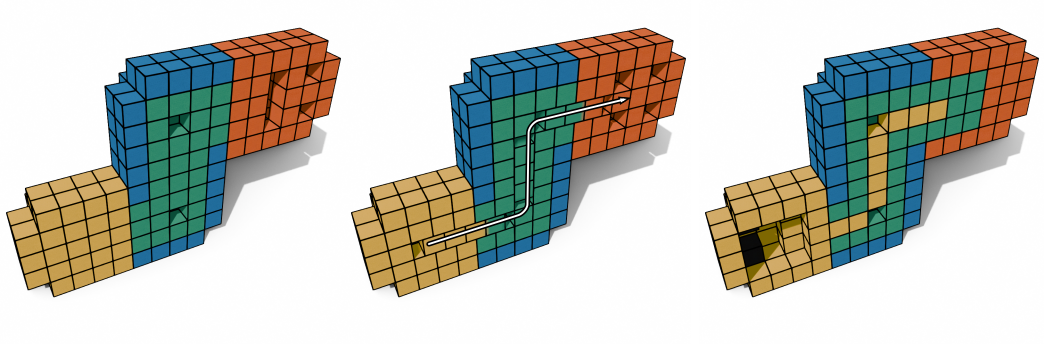


图 28 用于填充新前沿单元格的调度的二维表示。

完成后，新前沿和旧前沿都已被完全占据。随后我们可以采用类似命题25的论证，说明旧前沿的八个外层模块都是空闲的：此时新前沿是二顶点连通的，因此不会导致旧前沿中的任何模块变为非空闲。为了封闭新头部区段的外壳，这八个模块只需向外移动并占据那些单元格。再从头部区段内部最多调动四个额外模块，即可填满剩余的四个位置，封闭外壳。

完成这些变换后，新头部区段中最多会缺少十二个内部模块。为了恢复有效蛇形构型，我们利用脊柱路径主顶点处的自由空间，在构型内转弯，将适当数量的模块从尾部区段“传送”到新头部区段。由于头部缺少的模块数量始终有限，因此只需 $\mathcal{O}(1)$ 次变换。

◀



(a) (b) (c)

图29 为了像 (a) 所示那样填满头部的空单元格，我们沿着脊柱路径 (b) 从尾部推送模块，直到所有头部内部单元格都被占据，见 (c)。 e

▶ **引理 11** (⋆)。**给定一个蛇形配置 (**P)，可以通过 $O(1)$ 次变换交换头顶点和尾顶点的角色。

证明。 为此，只需将内部模块从头部传送到尾部区段，直到尾部区段填满。如果尾部区段大于头部区段，则可能需要在传送前将头部区段与相邻主顶点的区段合并。

无论如何，头部区段和尾部区段的体积都是恒定的，因此如图29(b)所示，沿脊柱路径逐个传送模块，速度足够快。◀

▶**引理 13** (⋆)。 设 S 是一个蛇形配置，其脊柱路径 P 的尾部区段中有一个空单元格，且 $v \in P$ 是脊柱的一个主顶点。通过 $O(1)$ 次变换，可以将一个自由模块 $m \in N_2^*[v]$传送到其内部，从而获得蛇形所有权。

证明。 如果包含 m 的单元格位于蛇形结构的单元空间内，即 m $\in S(P)$，我们只需将其状态从持有更新为已拥有。因此，假设 m $\notin S(P)$。

我们执行常数次分叉/合并和推拉操作。设 v_h，……，v，……，v_t为脊柱路径 P，按头部到尾部的顺序排列。首先在 v 处分叉蛇形结构，得到脊柱为 $P\prime := v$，……，v_t 的蛇形结构。由于 m $\in N_2^*[v]$，只需两次推操作即可变换分叉后的蛇形结构，使其包含 m。我们反转所得蛇形结构的方向，将内部模块从修改后的头部传送到位于 v_t 的原始尾部区段。朝向 v_t执行常数次操作后，我们便能将分叉后的蛇形结构的脊柱与原始蛇形结构重新对齐，并执行合并。◀

▶ **引理 14** (⋆)。 设 m_1，……，m_ℓ 为 T_s 中按 *DFS* 顺序排列的模块。包含 m_1 的蛇形配置 S 可以通过 $O(\ell)$ 次变换取得整个序列的所有权。

证明。 算法 1 中的伪代码是对蛇形结构如何实际遍历树的高度简化表示。

具体而言，我们不再只跟踪一条脊柱路径 P，而是跟踪一棵树 $\mathcal{P}$，它表示由于分叉而存在的脊柱路径的并集。这意味着，在任意时刻，我们当前考虑的蛇形结构的脊柱都由 $\mathcal{P}$ 中的一条路径定义。在算法首次调用之前，我们有一个脊柱为 P 的有效蛇形构型，即 $\mathcal{P} = P$。

设 $v_h \in \mathcal{P}$ 是当前选定的头顶点。根据算法，我们需要将其从一个元胞的中心延伸到相邻的元胞。由于每次推操作都需要一条特定的脊柱来传送模块，我们在$\mathcal{P}$ 中选取一条极大路径 P，该路径从 v_h

出发并通向树的一个叶顶点 v_t。如果所选脊柱的尾部区段已耗尽，也就是说，其中的模块数量不足以填满头部内部，我们可以执行拉操作，将该脊柱路径缩短一个单位，并从树$\mathcal{P}$中移除 v_t。由于我们为脊柱选取的是一条极大路径，因此这不会导致树断开。

在每次推操作之前，按照上述描述重新评估树，我们就总能找到一条脊柱，沿其执行算法第 4 行和第 7 行中的推拉操作；而分叉只需选择极大路径即可。

由于序列中的相邻模块面相邻，根据归纳法可知，每当从输入 DFS 序列中移除一个模块 m_i 时，它的后继模块 m_{i+1} 就位于与脊柱树 $\mathcal{P}$ 中某个顶点相邻的元胞中。因此，我们能够找到一个头顶点，以执行下一次推操作，从而到达该元胞。

最后，DFS 输入序列的长度为$\mathcal{O}(|T_\mathsf{s}|)$，而在从序列中移除下一个模块之前，我们只需要执行常数次推拉操作，因此最终模块将在$\mathcal{O}(\ell) = \mathcal{O}(|T_\mathsf{s}|)$次变换中被消耗。◀

C.4 缩放紧凑构型之间的重构

本节将证明，任意一个元模块紧凑构型都可以在$\mathcal{O}(A+h)$ 时间内变为另一个，其中 A 和 h 分别是两个构型之一沿某个坐标轴的投影面积和延展范围。

▶**引理15** ($\star$)。 任意两个元模块紧凑构型都可以通过完工时间为 $O(A+h)$ 的原位调度相互转换，其中 A 和 h 分别是这两个构型沿某个坐标轴 $\lambda \in \{x, y, z\}$ 的最大投影面积和最大延展范围。

为此，我们首先在给定紧凑构型的包围盒内创建一个长方体。我们的方法借鉴了二维情形中的思路，从而能够高效地并行化每一步。

▶**引理26**。 任意一个5倍缩放的紧凑构型 C，都可以通过完工时间为 $O(A+h)$ 的调度，转换为沿所选坐标轴 λ 投影面积大致相同的长方体。

证明。 不失一般性，假设 $\lambda = z$，其他情况均对称。为方便起见，令 $A = \mathbb{A}(z[C])$，并令 h 表示 C包围盒的垂直延展范围。

首先，我们确定一个面积为 $w \times d \approx A$ 的矩形 R，使其以原点为锚点并完全包含在 $z[C]$ 的包围盒内。如果 A 没有能实现精确匹配的整数因子，我们仍可假设 $A \geq (w-1)d$ 且 $A \geq (d-1)w$。

现在，我们反复对 C 进行变换，每次仅使用常数次变换，将其在 R 外的投影面积减少 25 个单位，即一个元胞。考虑 R 之外按 xy 取极大的已占用二维元胞 c。对于投影到 c 上的三维列中的每个元模块，我们都能在 xy 平面内找到一条 L 形路径，将其连接到 R 内的一个未占用三维元单元；这本质上是一个二维问题，可由 [3, 引理 15] 得出。然后，我们可以为每个三维元单元使用平行隧穿路径，仅通过常数次变换便将其传送到目的地。

由于投影与 R 的对称差至多为 A，经过$\mathcal{O}(A)$次变换后， C的投影将完全包含在 R 中。

为了将所得构型平衡为实际的长方体，我们先对所有 y-切片执行逐切片平衡操作，再对所有 x-切片执行该操作，分别将每个切片变为宽度为 w 或深度为 d 的矩形。对于每个坐标轴，这需要 $w+h$ 或 $d+h$ 次变换。

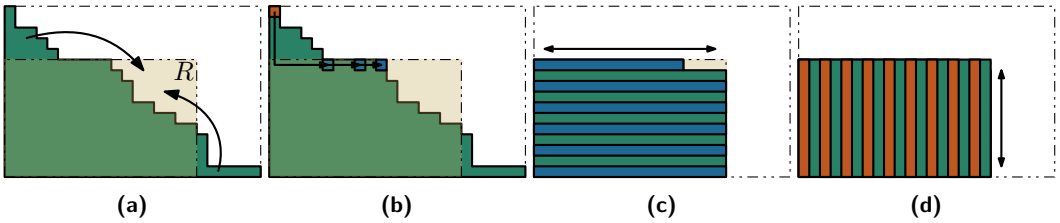


图30 通过以下方式，我们在所选矩形 R （黄色）内得到一个长方体：先（b）仅校正投影，然后沿（c） x-轴、再沿（d） y-轴进行垂直切片，并行完成。

结果是一个长方体构型，也就是说，它沿 z 坐标轴的投影为矩形，并且每个 z 列中的模块数量最多相差一个。◀

▶**观察27**。 设 B_1 和 B_2 是锚定于原点的两个轴对齐包围体。那么，或者（i）其中一个完全包含另一个，或者（ii）存在一个轴 $\lambda \in \{x, y, z\}$，使得 B_2 沿 λ 的延展范围大于 B_1的延展范围，并且 B_2 的延展范围沿另外两个坐标轴中的任一个都不大于 B_1 的延展范围。

▶**引理28**。 设 C_1、 C_2 是各含有 n 个模块的两个长方体构型。对于某个坐标轴 λ，存在一个由 $O(A + h)$ 次变换组成的调度，可将 C_1 变换为 C_2，其中 $A = \max(\mathbb{A}(\lambda[C_1]), \mathbb{A}(\lambda[C_2]))$ 是投影面积的最大值，而 h 是两个构型的包围盒沿 λ-轴的最大延展范围。

证明。 不失一般性，假设 $\lambda = z$ 并且 C_1 沿此坐标轴的延展范围较小，但沿 x轴和 y轴的延展范围均较大。令 $h_1 < h_2$ 分别表示C_1 和 C_2 沿 z轴的延展范围，如图31所示。

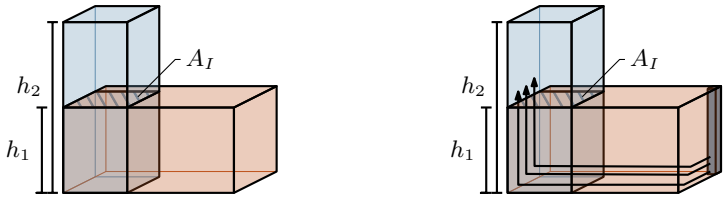


图31 我们在所有层上并行应用二维算法，基本上就能实现这一点。

根据引理26，我们可以假设 C_1 中的所有 z列高度相同（相差至多一个单位）。只需证明，经过常数次变换，我们就能做到以下任一项：(a) 将一个完整的 z模块列从 C_1 移入 $C_2 \setminus C_1$，或 (b) 填满体积 $C_2 \setminus C_1$ 的一个完整 z切片。

令 A_I 表示两个投影的交集面积；我们区分两种情况。如果 $A_I > h_1$，则可以很容易地证明情况 (a) 成立：我们可以在常数时间内将 C_1 中某个选定列里的所有模块“传送”到 C_2的包围体中。为此，我们在每个模块唯一的层中选取一条 L形路径，使其通向两个包围体的交集，然后再笔直向上穿入未占用的体积中。为避免碰撞，我们按如下方式嵌入这些路径。不失一般性，设 A_I 是顶点为 $(0, y, 0)$，$(x, y, 0)$，(x, y, z)，$(0, y, z)$ 的矩形，我们的目标是用列 $X \subset C_1$中的模块将其填满。将 X 中最低的模块分配到一条通往位置 $(0, y, 0)$ 的 L形路径，第二低的模块分配到位置 $(0, y, 1)$，依此类推，直到分配给位置 $(0, y, z)$ 的模块。接着将下一个模块放在位置 $(1, y, 0)$，然后以类似方式继续填充这一行。不难看出，这组路径互不相交，并且每条路径

最多有三次转弯。因此，沿着这些路径推动模块，就能“瞬间”移走选定的列，并将所有模块传送到所需的包围体中。

如果 $A_I < h_1$ 属于情况 (b)：我们选取该列的一个子集，其高度恰好为 A_I，并且只使用该列的这一部分。对于路径，仍然适用相同的技术；我们将对称差的一个 z 层完全填满。我们可以重复这一过程 A_I/h_1 次，以便在该列为空之前填满对称差的 $\lfloor A_I/h_1 \rfloor$ 层。

由于情况 (a) 和 (b)，我们需要 $\max(A, h_2)$ 的时间，将 C_1 中对称差的所有模块转移到 C_2 中的对称差，即将 C_1 重新配置为 C_2。◀