arXiv:2305.01015v1 [cs.CG] 1 May 2023

模块化机器人的碰撞检测——容易发生碰撞，却难以避免

SiddharthGupta ！ 华威大学，英国**MarcvanKreveld** ！ 乌得勒支大学，荷兰**Othon Michail** ✉利物浦大学，英国**Andreas Padalkin** ✉帕德博恩大学，德国

摘要

我们研究模块化可重构机器人的几何碰撞检测问题。假设节点（模块）是连接在网格上的正方形，我们探讨执行一组扩张和收缩操作时，判定是否可能发生碰撞或能否避免碰撞的复杂性。我们研究离散时间模型和连续时间模型，并允许将操作耦合为一个并行组。在具有 n 个节点的模块化机器人连续时间模型中，根据是否存在耦合操作及其类型，我们用于判定是否可能发生碰撞的算法分别可在 $O(n^2 \log^2 n)$ 时间、$O(n^2)$ 时间或 $O(n \log^2 n)$ 时间内运行。对于判定能否避免碰撞，我们证明，在离散时间模型中，一个限制非常严格的版本已经是 NP完全问题；而在连续时间模型中，同一问题可在多项式时间内解决。一个限制较少的版本在连续时间模型中是 NP困难问题。

2012年ACM主题分类 计算理论 → 计算几何；计算理论 → 算法设计与分析

关键词和短语 模块化机器人，碰撞检测，计算几何，复杂性

数字对象唯一标识符 10.4230/LIPIcs...

致谢 作者感谢所有参加贝尔蒂诺罗分布式几何算法研讨会的与会者，特别感谢 Peyman Afshani 提出了在不存在耦合的情况下检测碰撞的 $O(n \log^2 n)$ 时间复杂度解决方案。我们感谢 Irina Kostitsyna 和 Christian Scheideler 组织此次研讨会，同时也感谢后者提出了碰撞检测问题。最后，我们感谢 Jesper Nederlof 提出的一些有益见解。

1 引言

模块化可重构机器人技术及相关的可编程物质概念，关注的是由相互连接的基本实体组成的系统，这些实体称为模块。模块集合可以协调其有限的通信、计算、感知和局部驱动能力，以完成复杂的全局任务。模块通过一组可执行的机械操作来实现局部驱动。操作通常涉及执行该操作的模块，以及其局部邻域中的模块。此类操作包括推、拉、扩张、收缩、加倍和旋转。有些系统和模型采用的操作能够直接影响全局结构。当多个此类操作同时发生并相互重叠时，尤其如此。

局部操作能够对机器人结构产生全局影响，这是一把双刃剑。一方面，这是一种方便的并行性，可以让全局结构变化

©Gupta、van Kreveld、Michail、Padalkin；依据知识共享署名4.0国际许可协议授权。莱布尼茨国际信息学会议录，达格施图尔城堡——莱布尼茨信息学中心，达格施图尔出版社，德国

LIPICS

更快发生。另一方面，如果协调不当，就可能造成结构的轻微违规，甚至导致结构彻底失效。下文中，我们将所有结构违规和失效统称为碰撞。在单个模块上执行时能够对整个结构产生全局影响的操作，有时称为线性强度操作。

从理论角度看，已有多篇论文研究了此类操作的积极作用，涵盖不同的底层模型和操作类型。在一系列论文中，Aloupis 等人 [5, 4, 3] 研究了一种称为晶体机器人的机器人系统模型 [17]。晶体模型的二维版本将模块表示为二维网格上的正方形，构成由相邻模块连接而成的连通形状。每个模块都可以通过将其中一个面向外伸出一个单位，并在稍后将其缩回，从而进行扩张和收缩。由于模块彼此连接，一个模块的扩张或收缩可能使多达线性规模的组件发生移动。在 [5], Aloupis 等人为晶体模型提出了一种通用集中式重构算法，对于任意一对模块数量相同的连通形状 S_I，S_F，该算法都能在 $O(\log\ n)$ 个并行时间步内将 S_I 变换为 S_F，执行 $\Theta(n\log\ n)$ 次单独操作。

在 [19], Woods 等人提出了 nubot 模型，其灵感来源于分子的可编程自组装，例如 DNA 链。在该模型中，模块表示二维三角网格上的单体。该模型包含多种不同类型的操作，例如模块的插入、删除和旋转。操作引起的运动会通过连接传播到形状中更大的部分。如果操作的全局效果会违反连接的刚性，则假定该操作会被取消。他们的主要成果是一种分布式异步算法：从单个模块开始，该算法能够在期望的、关于 n为多对数级的并行时间步数内，形成大小为 n的任意连通二维形状和图案。

Almalki和Michail [1], 基于 [19] 的插入操作以及Mertzios 等人 [16], 关于图上的生长过程的研究，考察了仅使用增长操作时，哪些形状族能够在关于其大小为多对数级的时间内生长出来。他们在二维方格网格模型中进行研究，并考虑了每个时间步必须满足的、关于操作耦合的不同要求。他们对操作进行了定义，使得碰撞永远不会发生。他们给出了从形状 S_I （可能是单个模块）生长出形状 S_F 的集中式算法；对于大类形状，这些算法产生了多对数级并行时间步调度。

Derakhshandeh等人 [12]提出的变形虫机器人模型，以及其近期的规范扩展 [8]，是另一种主要考察模块扩张与收缩操作的模型。模块在二维三角网格上运行，并通过将模块头部向空白空间扩张来实现重构，随后再将尾部向头部收缩。该模型中的形状形成算法通常设计为并行执行操作，但每项操作只影响其周围的局部区域，而不会影响形状的更大部分。参见 [7], ，其中有一章近期内容介绍了该模型中的主要算法进展。近期，Feldmann等人 [14] 提议在该模型中加入线性强度操作，但将这类扩展的具体细节留待未来研究。

在通过折叠 [15]实现的可编程物质中，也存在避免形状的大型运动部件之间发生碰撞的必要性，以及与之相关的算法问题 [11]。在这些模型中，目标是从足够大的表面（例如一张纸）开始折叠多面体复形，并在此过程中避免自相交。协调运动规划是另一个与我们的工作有一定相似之处的研究方向。在该领域，目标是在避免机器人碰撞的同时，尽可能快速地将机器人群体重新配置为目标构型 [18, 10]。Almethen 等人 [2]提出的线推模型是另一种模型，其中单个操作对形状的影响最多可以是线性的。

显然，大多数研究都将注意力局限于那些可以安全并行执行的操作。这些操作要么是不会发生碰撞的线性强度操作，要么仅影响其周围局部区域的操作。本文明确提出了一个算法问题，即确定一组操作在何种情况下可能导致碰撞，以及在何种情况下可以避免碰撞。具体而言，给定一个形状以及作用于该形状的一组线性强度操作，我们旨在提出集中式算法，以计算这些操作（集合）的调度，使其能够实现以下目标：（i）导致碰撞，或（ii）避免碰撞。前一个子问题的研究动机源于异步分布式算法，在此类算法中，操作的任何一种可能的交错执行都可能是模块实际采用的执行方式；后一个子问题则源于设计高效重构算法的需求，即通过算法本身避免碰撞，而不是将碰撞规避机制内置于模型之中。据我们所知，本研究是首个明确探讨此类问题的研究。

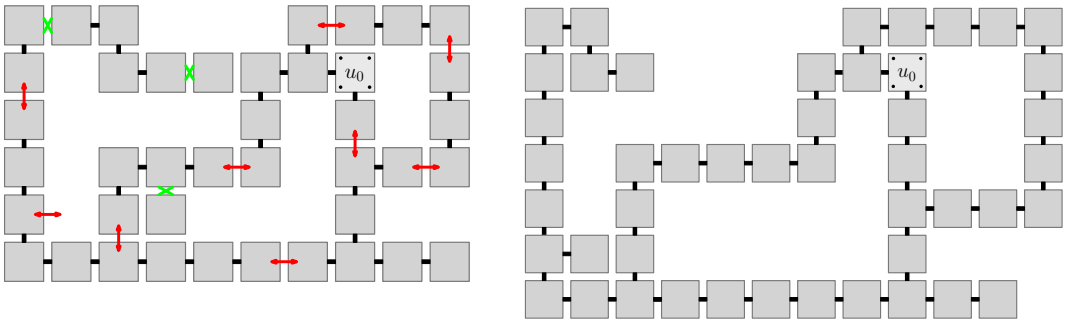


图1 对一个形状应用一组无碰撞的并行操作。通常，形状的边集以节点之间的黑色边表示，而 u_0 表示锚点。红色双箭头表示扩张，绿色 X 表示收缩。两个节点之间的任何操作都应视为沿一条边进行；为使插图简洁，图中未绘出这条边。出于同样的原因，节点绘制得比实际尺寸略小。

研究方法。 我们选择在二维方格网格模型中研究这些问题，其中模块（下文称为节点）是占据网格中不同网格单元的单位正方形。这样的选择使 Aloupis 等人 [5] 提出的二维晶体模型以及 Almalki和Michail[1] 提出的生长模型与我们的模型最为接近。节点可以与部分相邻节点相连，且这些连接始终形成一个初始连通形状 S。我们不允许在节点之间建立新的连接。所考虑的操作为扩张和收缩。这些操作也可以视为 amoebot 模型 [12]中扩张和收缩操作的线性强度扩展。我们问题的输入包括一个连通形状S、对 S 进行的不带重数的操作分配，以及一个操作的耦合划分。耦合划分指定哪些操作将并行执行。示例见图1。

结果与概述。第2节介绍我们研究碰撞问题所采用的模型。具体而言，我们说明假设的输入、操作及其耦合、碰撞和时间模型。随后给出问题定义。第3节首先介绍用于检测在连续时间模型中是否可能发生碰撞的算法，在该模型中，操作可以随时开始。当未指定操作之间的并行关系时，我们给出一个运行时间为 $O(n\log^2 n)$的算法。当指定某些操作并行执行时，我们则根据并行程度，介绍运行时间为 $O(n^2)$或 $O(n^2\log^2 n)$的算法。

接下来，我们给出离散时间模型中的多项式时间算法。在该模型中，任何操作或耦合操作组都必须完成其动作后，下一个操作才能开始。第4节讨论如何确定能否在避免碰撞的情况下执行所有操作。我们证明，在离散时间模型中，一个受限版本的问题已经是NP完全的，而在连续时间模型中，该问题可以在多项式时间内求解。一个限制稍少的版本在连续时间模型中仍然是NP困难的。NP困难性结果要求指定某些操作并行执行。

2 模型

我们假设使用一个二维方格网格，其中每个网格单元都有整数坐标（x，y）。节点（模块）占据网格单元，从而定义一组被占据的整数点，且任意两个节点不会占据同一网格单元。我们将每个节点 $u = (u_x, u_y)$表示为一个正方形，其大小与网格中的网格单元（u_x，u_y）相同且完全对齐。一个形状 $S = (V, E)$是由节点V及其连通关系构成的配置，以 E表示。只有正交相邻节点才能连接，但相邻节点不一定相连。我们用 n 表示 $|V|$，并始终将讨论范围限定为连通形状。

操作与碰撞。一般而言，对一个形状 S 执行一个或多个操作，要么会导致碰撞，要么会产生一个新形状 S'。碰撞分为两种：节点碰撞和环碰撞。鉴于这里的所有碰撞都是连通形状的“自碰撞”，不失一般性（下文简称“w.l.o.g.”），我们可以假设存在一个锚点节点 $u_0 \in V$ 保持静止，而其他节点相对于它移动。我们先从较简单的情况开始：形状是一棵树 $T = (V, E)$，此时不存在环碰撞；随后再将讨论推广到任意连通形状 S。

我们首先定义单次扩张和收缩操作；[1] 见图1。扩展操作作用于一对相邻的整数点 uv，其中满足以下条件之一：（i） $u \in V$和 $v \notin V$；或（ii） u、$v \in V$和 $uv \in E$。剩下的情况是 u、$v \in V$但$uv \notin E$，这会立即导致碰撞。在情况（i）中，扩展会在空单元格 v 处生成一个节点，并将其连接到 u。在情况（ii）中，不失一般性，假设 u 在 T 中比 v更靠近 u_0。令 T（v）表示以 v为根的 T 子树。随后，扩展会在u 和 v之间生成一个节点，并将其连接到两者；同时，沿平行于 uv的轴将 T（v）向远离 u 的方向平移一个单位。在这两种情况下，新节点起初是一条单位长度线段，随后扩展成一个单位正方形。收缩操作作用于一对节点 $uv \in E$、v，它们是距离锚点最远的节点。该操作将 v 与 u 合并，同时将 T（v）朝 u 方向平移一个单位，而v 则收窄为一条单位长度线段。在这两类操作中，如果 T（v）平移后有两个节点占据同一单元格，就发生了碰撞。我们将这种碰撞称为节点碰撞，并更一般地将其定义为任意时刻任意两个节点的区域存在非空交集。否则，就得到了一个新的树 T'。

我们假设每个节点最多只参与一次操作。

耦合。设 Q 为一组要并行应用于连通形状 S的操作，每项操作作用于一对不同的节点，或一个节点和一个空单元格。我们称这样的一组操作为耦合，其中的操作称为耦合的或并行的。我们假设 Q 中的所有操作均同时应用，具有相同的恒定执行速度，且其持续时间为一个时间单位。

[1] 我们认为，我们的定义和技术可以扩展到扩张和收缩的其他形式——包括操作可以逆转的情况——以及三角网格等不同几何结构。

设 $T=(V,E)$ 为一棵树，$u_0 \in V$ 为其锚点。令 u_0 为 T 的根。我们希望确定并行应用 Q 中的操作后，每个 $v \in V \setminus \{u_0\}$ 的位移。由于 u_0 静止不动，且每项操作都会平移一个子树，因此只有位于唯一 u_0v 路径上的操作才会影响 v 的位移。具体而言，每项这样的操作都会使 v 的运动向量 $\vec{v}$ 增加以下某个单位向量：$\mathrm{h}-1, \mathrm{0i}$、$\mathrm{h}0, -\mathrm{1i}$、$\mathrm{h}+1, \mathrm{0i}$、$\mathrm{h}0, +\mathrm{1i}$。此外，对于每个朝空单元格扩展的节点 $u \in V$，我们都会添加一个新节点 v，并为其指定相应的单位运动向量 $\vec{v}$。我们可以利用运动向量集合来确定任意两个节点的轨迹是否会在某处发生碰撞。现在，设 S 为任意一个至少包含一个环的连通形状，u_0 为其任意一个节点，并将其作为锚点。此时，S 上的一组操作 Q 要么会导致环碰撞，要么其效果实质上等同于将 Q 应用于以 u_0 为根的 S 的任意生成树。设 u、v 为环上的任意两个节点。若 p_1 和 p_2 是该环的两条 uv 路径，则必须满足 $\vec{v}_{p_1} = \vec{v}_{p_2}$：沿路径 p_1 和 p_2 的位移向量相等。否则，我们就无法保持环上的所有节点或边。此类违规情况称为环碰撞。我们称不会导致任何节点碰撞或环碰撞的一组操作为无碰撞。

离散时间与连续时间。 我们考虑两种不同的操作调度模型。在离散时间模型中，每个操作或耦合操作都在不同的整数时刻开始，并且恰好持续一个时间单位。换言之，除非两个操作相互耦合，否则不会有两个操作同时处于执行状态。在连续时间模型中，我们不作开始时间必须为整数的假设。操作可以在任意时刻开始，其执行时间可以重叠。耦合操作同时开始并同时结束。对于这两种时间模型，我们都假设每个操作恰好需要一个时间单位完成，且执行速度恒定。在离散时间模型中，只有操作（单独或耦合）的顺序会影响是否发生碰撞。在连续时间模型中，操作的确切开始时间会产生影响。

问题定义。 现在我们定义所考虑的问题。给定一个形状 S 以及在 S 上的操作分配（其中任一节点至多涉及一次），S 上操作的耦合划分是一个集合族$\{Q_1、Q_2、\ldots\ldots、Q_k\}$，其中每个 Q_i（可以是单元素集合）表示一个应并行执行的操作子集。

碰撞调度。 给定来自某个形状族的形状 $S=(V、E)$，以及在 S 上的操作耦合划分 $\{Q_1、Q_2、\ldots\ldots、Q_k\}$，请判断是否存在一个开始时间 $t_0(Q_i) \in \mathbb{R}$，使得对于每个耦合集合 Q_i，按照这些开始时间执行操作会导致碰撞。

无碰撞调度。 给定来自某个形状族的形状 $S=(V、E)$，以及在 S 上的操作耦合划分 $\{Q_1、Q_2、\ldots\ldots、Q_k\}$，请判断是否存在一个开始时间 $t_0(Q_i) \in \mathbb{R}$，使得对于每个耦合集合 Q_i，按照这些开始时间执行操作时不会发生碰撞。

这些问题的离散特殊情形，即离散碰撞调度和离散无碰撞调度，分别通过要求所有 $t_0(Q_i)$'s 均为互不相同的整数而得到。

碰撞调度的 3 种算法

本节将介绍用于判定某个连通形状在某种操作调度下是否会发生碰撞的算法。我们将分别讨论无耦合的情况，以及存在各种形式耦合的情况。首先讨论 S 是树的情况，然后扩展到一般连通图上的碰撞调度问题。

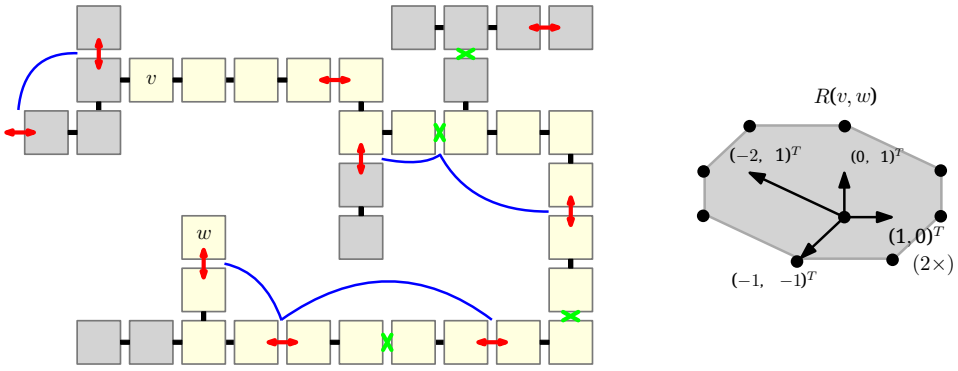


图 2 一个具有操作以及 v 和 w 的形状。只有两端都位于 vw 路径上的操作，才会影响 w 相对于 v的移动方式。耦合操作以蓝色弧线表示。该路径总共隐含了五个向量，其中两个相同。 w相对于 v 的对应区域 $R(v,\ w)$是右侧所示的zonotope。zonotope中的（2×）标签表示向量（1，0）^{T}由该路径隐含了两次。

3.1 树的连续和离散碰撞调度

我们假设 S 的拓扑结构是一棵树。首先介绍一种通用方法，用于判定是否可能发生碰撞，该方法适用于存在耦合的情况。当所有耦合均为固定大小，或每个耦合仅涉及水平方向或垂直方向时，该方法的效率会略高。然后介绍一种适用于不存在耦合情况的改进算法。

令 v 为 S中的任意节点。我们将给出一种算法，用于判断 v 是否会与其他任意节点 w 发生碰撞。请回忆， v 与 w之间恰好存在一条路径，只有该路径上的操作和耦合才决定 v 与 w 是否会发生碰撞。假设 v 保持静止，该路径上的耦合操作允许 w 沿着具有整数坐标的向量$\langle i,\ j\rangle$ 进行平移。让所有节点依次充当 v，即可得到一个完整的算法。

从 v开始，我们遍历 S 并访问所有其他节点。每当我们访问并处理节点 w'时，我们已经访问了从 v 到 w'的路径上的节点，特别是这条路径上 w' 的邻居 w 。该算法的核心思想是在遍历 S 、逐个访问节点时，维护由操作和耦合构成的向量，并保留其重数。

令 Vec(v,w) 表示描述 v 和 w 之间独立操作与耦合的向量多重集（见图 2），并令 $\mathcal{T}(v,w)$ 表示一棵树，该树将 Vec(v,w) 按角度排序后存储在叶节点中。我们通过在 $\mathcal{T}(v,w)$ 的每个内部节点和叶节点中存储其下方叶节点中所有向量（包含其重数）的向量和，来扩展该树。

在 S 中增加一个遍历步骤，从 w 到 w'，意味着我们必须更新 T (v,w)，以获得 T (v,w')：(i) 如果 w 和 w' 之间的边不是一个操作，则 T (v,w') = T (v,w)。(ii) 如果该边是一个操作，且迄今为止的路径上没有耦合，则我们从 h-1, 0i、h0, -1i、h$+1$, 0i、h0, $+1$i 中得到一个新向量，并将其插入 T (v,w)（也可以通过增加重数而非进行叶节点插入），从而创建 T (v,w')。(ii) 如果该边是一个操作，且与 v 和 w之间的一个或多个操作耦合，则通过向一个向量添加一个额外操作来修改该向量。相应的向量沿 x- 或 y-方向变化 ±1 。该向量可能变为零向量，此时我们将其删除。否则，我们执行一次删除和一次插入来改变该向量。

树 $\mathcal{T}(v,w)$ 是节点w 相对于 v 的可达区域 $R(v,w)$ 的隐式表示。在连续时间模型中，可达区域是一个包含节点 w 本身的凸区域。它是向量 Vec(v,w) 的 *zonotope* ，其边的数量是向量不同方向数量的两倍（我们注意到，如果

存在耦合且我们使用离散时间模型，则 (,) 不是凸的）。在连续时间模型中， w通过在合适的时间开始操作，总能与内部的任意单元相交。

下一个引理表明，我们可以高效地查询搜索树 $\mathcal{T}(v,w)$，以在给定的查询方向上找到 $R(v,w)$ 的一个极端点，并测试节点 v和 w是否相交。

▶ **引理 1**。 我们可以在 $O(\log n)$ 时间内，沿给定的查询方向 $\vec{\rho}$找到 $R(v,w)$ 的一个极端点，并在 $O(\log^2 n)$ 时间内测试 v 和 w 是否相交。

证明。令 $\vec{\rho}$为任意向量；我们只使用它的方向。下面说明如何利用 T (v,w) 找到 $R(v,w)$ 中沿 $\vec{\rho}$方向的极端顶点。对于这样的顶点 u，过 u 作一条垂直于 $\vec{\rho}$ 的直线，则 $R(v,w)$ 的内部将严格位于直线的一侧。为找到这个顶点，我们将所有与 $\vec{\rho}$的点积为正的向量相加。对 T (v,w) 的增强使我们能够通过相加存储在内部节点或叶节点中的 $O(\log n)$ 个向量来找到该顶点。我们沿着 T (v,w) 中的搜索路径向下遍历，使用垂直于 $\vec{\rho}$的两个方向，即 $\vec{\rho}-90°$ 和 $\vec{\rho}+90°$。假设我们分别到达叶节点 ν 和 ν'。对于两条通往 ν 和 ν'的路径之间的所有最高节点，我们将存储在这些节点中的向量和相加。（请注意，由于方向具有循环顺序，我们可能还需要左侧路径左边和右侧路径右边的节点。我们忽略这一完全类似的情况。）如有需要，我们还会将 ν 和 ν' 中的一个或两个向量（及其重数）加入总和。该总和可在 $O(\log n)$ 时间内给出沿 $\vec{\rho}$方向的 $R(v,w)$ 的极端顶点 u。当我们将 $\vec{\rho}$稍微转动时，u仍将是极端顶点，直到方向变化足够大，使得与 $\vec{\rho}$ 的点积为正的向量集合发生变化。当 Vec(v,w) 中存在方向为$\vec{\rho}-90°$ 或 $\vec{\rho}+90°$ 的向量时，就会发生这种情况。这样的向量存储在 ν或 ν' 或相邻的叶节点中。通过检查这些向量，我们可以在 $O(\log n)$ 时间内确定 $R(v,w)$ 的相邻极端顶点 u'，进而确定一条界定 $R(v,w)$ 的边 uu'。

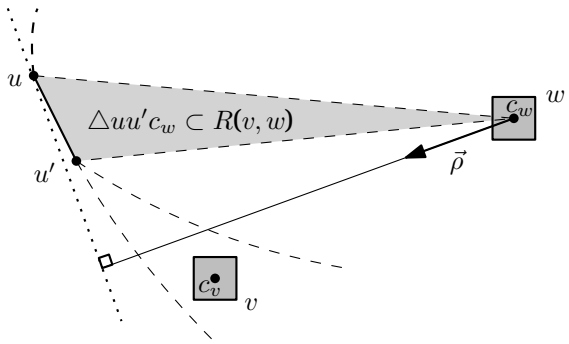


图3 情形（iii）示意图，其中 u 是 $R(v,\ w)$ 沿 $\vec{\rho}$方向的极端顶点，而 u和 u'的位置尚不能判断 $c \in R(v,\ w)$的情况（虚线曲线表示 $R(v,\ w)$沿逆时针方向的可能延续）。此处我们必须进一步沿逆时针方向搜索，方法是选择另一个相对于当前 $\vec{\rho}$正向旋转的 $\vec{\rho}$。

我们将使用沿方向 $\vec{\rho}$ 对 $R(v,\ w)$ 的极端顶点进行查询，以判断 v 是否与 w 能够到达的区域相交。我们首先说明如何判断节点 v 的中心点 c_v 是否位于 $R(v,\ w)$ 内；该区域是节点 w 的中心点 c_w 能够到达的区域。然后，我们将此方法应用于网格单元本身。我们通过找到 $R(v,\ w)$ 的边 uu'来判断 $c_v \in R(v,\ w)$ 是否成立；其中 $\overrightarrow{c_w c_v}$位于 $\overrightarrow{c_w u}$和 $\overrightarrow{c_w u'}$之间，见图3。找到这条边后，我们就可以检验 $c_v \in \triangle uu'c_w$；若成立，则发现了碰撞，否则没有发现碰撞。

XX:8 模块化机器人碰撞检测

为了找到 $R(v,w)$ 的边 uu'，我们执行二分搜索，首先取 $\vec{\rho}$对应的角度区间，其逆时针边界为 $\overrightarrow{c_wc_v}-90°$，顺时针边界为 $\overrightarrow{c_wc_v}+90°$。这两个边界分别指向 T (v,w) 中的叶节点。我们取中间位置的叶节点 μ，执行一次二分搜索。该叶节点 μ 对应从 w出发的方向 $\vec{\rho}$，这个方向给出 $R(v,w)$ 的一对极点 u 和 u'。有三种情况，见图3。(i) 如果 c_v 位于三角形 $\triangle uu'c_w$ 内，那么我们知道 c_v 位于 $R(v,w)$ 内，并且发现了一种可能的碰撞。否则，(ii) 如果与 c_w相比，c_v 位于经过 u和 u' 的直线另一侧，那么根据凸性，c_v 不可能位于 $R(v,w)$ 内，我们可以停止。否则，(iii) 我们必须检查比 $\vec{\rho}$更偏顺时针或更偏逆时针的方向。

通过这种二分搜索，我们最多需要 $O(\log n)$ 步来寻找极值点，直到出现 $\overrightarrow{c_wc_v}$ 位于 $\overrightarrow{c_wu}$ 和 $\overrightarrow{c_wu'}$ 之间的情况；此时我们便可以判断 c 是否位于 $R(v,\ w)$ 中（根据情形 (i) 和 (ii)）。

到目前为止，我们忽略了 v 和 w 是正方形而不是点这一事实。只需对 $R(v,\ w)$ 按固定幅度进行膨胀，即可轻松将这一点纳入算法。一个简单的方法是在树 $\mathcal{T}(v,\ w)$ 中加入四个向量 $(\pm1,\ 0)^T$ 和 $(0,\ \pm1)^T$ 作为额外向量，从而有效扩张可达区域。这样，我们就可以继续处理中心 c_v 和 c_w，而不是正方形 v 和 w。◀

对于完整算法，我们有 n 个起始节点 v，并对每个起始节点遍历 S。处理每个遇到的节点 w 时，在 w 区域中搜索 v 需要 $O(\log^2 n)$ 时间，更新 $\mathcal{T}(v,\ w)$ 以便为遍历 S 中的下一个节点 w' 做准备，则需要 $O(\log\ n)$ 时间。因此，整个算法的运行时间为 $O(n^2\log^2 n)$。当所有耦合的大小均为 $O(1)$，或所有耦合均为纯水平或纯垂直时，算法的分析很简单：区域 $R(v,\ w)$ 和树 $\mathcal{T}(v,\ w)$ 的复杂性均为常数，因此对数因子消失。运行时间可改进为 $O(n^2)$。

完全没有耦合时，我们采用一种不同的方法，从而得到更高效的算法。选择形状 S 的一个锚定节点 v，移除该节点后会形成若干子树（子形状），每棵子树的大小都不超过原形状的一半。这样的中心节点总是存在，且最多会形成四棵子树。注意，在一棵子树中的任何操作都只能影响该子树中节点的位置，不会影响其他子树中节点的位置。

我们将分别确定每棵子树中的所有节点可以位于何处。如果任意两棵子树的这些区域有重叠，那么我们就找到了一对分别位于不同子树中的节点，它们可能发生碰撞，问题也就得到了解答。如果某棵子树中的某个节点可以占据锚点 v 的位置，我们同样就得到了解答。否则，我们将在每棵子树中递归地解答这个问题（注意，同一棵子树中两个节点的区域相交并不能说明任何问题）。在递归步骤中，锚点各不相同，区域也各不相同。

对于子树 S_1 中的每个节点 w_1，考虑 v 与 w_1 之间路径上的所有操作。这些操作包括水平和垂直收缩与扩张。它们共同确定了 w_1 可能处于的所有位置。由于不存在耦合，这些位置必然构成一个矩形 $R(v,w_1)$。我们对 S_1 中的所有节点执行此操作，得到一组$\mathcal{R}$ “红色”矩形；对另一个子树 S_2 中的所有节点执行相同操作，得到一组$\mathcal{B}$ “蓝色”矩形。我们可以通过从锚点 v 进行树遍历，并维护从 v 出发路径上的操作，在 线性时间内计算出$\mathcal{R}$和$\mathcal{B}$。借助标准的平面扫描和线段树，我们可以在 $O(n\log\ n)$ 时间$\mathcal{R}$ 内判断其中是否有任何矩形与$\mathcal{B}$中的任何矩形相交 [9]。由于要对 S 的子树进行递归，我们总共花费 $O(n\log^2 n)$ 时间。

▶**定理 2.** 设 S 是由 n 个单位正方形节点组成的形状，并在相邻节点之间的边上定义操作，且 S 的邻接结构是一棵树。那么我们可以求解 碰撞调度

- 若存在耦合，则时间复杂度为 $O(n^2\log^2 n)$ ；若每个耦合的大小固定，或仅为水平耦合或
- 垂直耦合，则时间复杂度为 $O(n^2)$ ；若操作之间没有耦合，则时间复杂度为 $O(n\log^2 n)$ 。
-

我们也可以在多项式时间内求解离散碰撞调度。无耦合时的算法仍然正确，但存在耦合时则需要采用不同的方法。现在，我们给出一种算法，用于判断在离散时间模型中存在耦合时，是否可以相对于另一个节点 v移动 w ，使二者发生碰撞。回顾一下， v 和 w 之间路径上的耦合意味着一组向量 Vec(v,w)。与连续时间情形不同，存在耦合时， w 的可达区域 $R(v,w)$不再是凸集。

我们的方法是对经典动态规划方法的扩展，该方法用于在数值为有界整数的情况下求解子集和问题。两者主要有两点区别。首先，我们处理的不是整数，而是整数对（向量）。其次，我们需要处理节点 w 的移动在移动过程中与 v 发生碰撞的情况。

我们将 Vec(v,w) 中的向量按任意顺序排列为 $\tau_1,\ldots,\tau_m$，并且对于所有 $1\le j\le m$，省略向量 τ_j，将其余向量重新命名为 $\nu_1,\ldots,\nu_{m-1}$。我们将展示如何利用这些 ν 个向量的所有子集，生成 w 能够到达的所有单元格位置，然后测试 w 沿向量 τ_j 移动是否会与 v 发生碰撞。由于每个碰撞调度都必然以某个导致碰撞的向量结束，因此我们涵盖了所有可能性。

我们将一个状态（i,x,y）定义为一个布尔值：当且仅当 $\nu_1,\ldots,\nu_i$ 的某个子集，其第一部分的和等于 x，第二部分的和等于 y时，算法才应将其设为 True。我们将该状态存储在一个三维表中，其大小为 $O(n^3)$，其中 $m\le n$ ，且 x 和 y 的最大值为 n。

按照动态规划的方式，我们根据第一索引较小的表项来定义 $(i+1,x,y)$。初始化时，将表中所有表项设为 False。然后令 (0,0,0) := True。令 $\nu_{i+1}=\langle\nu^x_{i+1},\nu^y_{i+1}\rangle$。

- 如果 (i,x,y) =True，则令 $(i+1,x,y)$:=True。如果 (i,x,y) =
- True，则令 $(i+1,\ x+\nu^x_{i+1},\ y+\nu^y_{i+1})$:=True。

填完表后，我们就知道最终移动前 w 在向量 τ_j 上的所有可能起始位置。我们逐一检验这些位置是否会与 v 发生碰撞。该算法总共需要$O(n^4)$ 时间。通过以下观察可以改进这一基本算法：

▶ **引理 3.** 集合 Vec(v,w) 至多有 $O(n^{2/3})$ 个不同的向量。

证明。 任意向量 $\langle i,\ j\rangle$ 在单次耦合中，至少需要 $|i|+|j|$ 个节点，位于 v 与 w 之间的路径上。要得到互不相同的向量，我们需要越来越大的耦合。显然，某一维的绝对值大于 $n^{2/3}$ 的向量至多有 $n^{1/3}$ 个。同时，两维绝对值都不超过 $n^{1/3}$ 的向量数量为 $4n^{2/3}$。◀

由于向量加法满足交换律，我们可以假设所有方向相同的向量都连续排列。这意味着我们可以排除的不只是一个向量 τ_j，而是所有相等的向量。最后，我们用该方向上的向量之和来检测碰撞。由于只有 $O(n^{2/3})$ 个不同的向量，我们只需运行 $O(n^{2/3})$ 次动态规划方法，运行时间为 $O(n^{11/3})$ ，以检测 v 和 w 是否相交。

此外，当耦合的大小为 O（1），或者只有水平和垂直方向的耦合时，向量方向只有 O（1）种，因此运行时间为 O（n^3）。这些运行时间是针对每对节点 v, w 而言的，因此总体上我们得到：

▶**定理 4**。 设 S 是由 n 个单位正方形节点组成的形状，操作定义在相邻节点之间的边上，且 S 的邻接结构是一棵树。那么我们可以求解离散碰撞调度

- 在 $O(n^{17/3})$ 时间内（如果存在耦合）；在 $O(n^5)$ 时间内（如果每个耦合具有固定大小，
- 或仅限水平方向或垂直方向）；在 $O(n\log^2 n)$ 时间内（如果这些操作之间不存在耦合）。
-

对于最后一个结果，我们可以使用连续时间模型的结果，因为不存在耦合时，离散情形和连续情形下的可达区域$R(v, w)$相同。

最后，我们注意到，当所有操作只有一个耦合时，碰撞调度与无碰撞调度相同，离散时间模型与连续时间模型也是如此。我们可以通过添加第三维来对该问题建模。每个节点在平面 $z = 0$ 中位于其起始位置，并在平面 $z = 1$ 中位于其目标位置（由某个锚点给定）。我们为每个节点构造一个水平截面为正方形的棱柱，将起始正方形与目标正方形连接起来。于是，判断是否发生节点碰撞归结为确定这些棱柱之间是否存在适当的相交。对于任意常数 $\epsilon > 0$，我们可以沿着其他棱柱的面上的所有棱柱边进行射线投射，在 [13] 的 $O(n^{3/2+\epsilon})$ 时间内解决该问题。

3.2 一般图的碰撞调度

回想一下，如果环上两个节点 u 和 v 之间的两条路径对于其中一个节点相对于另一个节点的移动位置存在分歧，那么这两个节点之间就会发生环冲突。对于环上包含单个操作的任何环，只执行该操作就很容易造成环冲突。更一般地说，水平操作必须处于能够抵消其对环造成影响的耦合中，垂直操作也同样如此。如果对任何环来说都不满足这一条件，就可能产生环冲突。是否可能发生环冲突，并不取决于我们检查其相对移动情况的那两个节点。环上的任意一对节点都会得到相同结果。

我们可以证明，如果耦合满足以下条件：图中的每个初等环 都不存在环冲突，那么整个图也不存在环冲突。因此，我们只需要检查$O(n)$个初等环。下面将对此作更精确的说明。

令 $S = (V, E)$ 为连通形状的连通图。初等环是双连通分量中有界面的外边界。我们通过移除图中的所有割点来得到这些初等环。剩下的是零个或多个分量，每个分量都是双连通的。嵌入中的每个有界面都由一个由节点构成的简单环围成。这些就是初等环。

▶**引理 5**。 对于任意连通形状 S，初等环不会导致环冲突，当且仅当所有环都不会导致环冲突。

证明。 我们只需证明，初等环之间不存在碰撞，就意味着任意环之间都不存在碰撞。

不难看出，在任何简单环中，水平操作都必须与该环中的另一项水平操作耦合。根据对称性，垂直操作也同样如此。同样显而易见的是，与割点关联的操作不会造成环

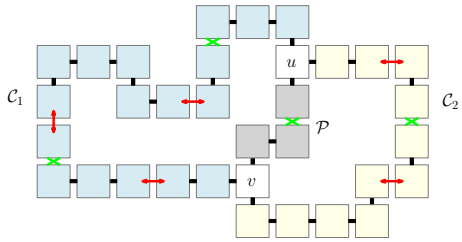


图4 一个简单环 $\mathcal{C}$ 在两个节点 u 和 v处分成 $\mathcal{C}_1$ 和 $\mathcal{C}_2$，并有一条内部路径 $\mathcal{P}$（包含三个节点）连接 u 和 v。

冲突，只会造成节点碰撞。后一个观察结果表明，我们可以将范围限制在双连通分量内。

设$\mathcal{C}$ 是一个双连通分量中的任意环，并假设该双连通分量的所有初等环都不会造成环碰撞。我们将通过对$\mathcal{C}$内部面数 f 进行归纳，证明$\mathcal{C}$ 也不会发生环碰撞。

若 $f = 1$，则$\mathcal{C}$ 是初等环，因此根据假设结论成立。设 $f > 1$，并假设内部有 $\leq f - 1$ 个面的所有环都不会造成环碰撞。在该双连通分量中取一条简单路径$\mathcal{P}$，且它位于$\mathcal{C}$内部；见图4。设 u，v 是$\mathcal{C}$上与$\mathcal{P}$的首、末节点相邻的两个节点。我们考虑 u和 v之间的三条路径：两条沿$\mathcal{C}$，另一条就是$\mathcal{P}$。设 u 和 v 分别为这些路径的起点和终点，这样，$\mathcal{C}$上的操作、$\mathcal{P}$上的操作，以及$\mathcal{P}$ 与 u 和 v 之间的操作，都恰好发生在其中一条路径上。将这三条路径称为$\mathcal{C}_1$, $\mathcal{C}_2$,和$\mathcal{P}'$。

根据归纳法可知，由$\mathcal{C}_1$ 和$\mathcal{P}$ 构成的环，以及由$\mathcal{C}_2$ 和$\mathcal{P}$ 构成的环，都不会造成环碰撞，因为它们内部的面少于 f 个。将 u 视为锚点，并考察$\mathcal{C}_1$ 上的操作如何移动 v。这由向量$\langle i, j \rangle$指定。由于由$\mathcal{C}_1$ 和$\mathcal{P}$ 构成的环不会造成碰撞，路径$\mathcal{P}'$ 也必须沿同一向量$\langle i, j \rangle$移动 v。对$\mathcal{C}_2$ 和$\mathcal{P}$重复这一论证，我们可以看到$\mathcal{C}_2$ 也必须沿$\langle i, j \rangle$移动 v。因此，环$\mathcal{C}$ 也不会造成环碰撞。对于任何能够确保较小的环不会造成环碰撞的耦合，这一论证都成立。◀

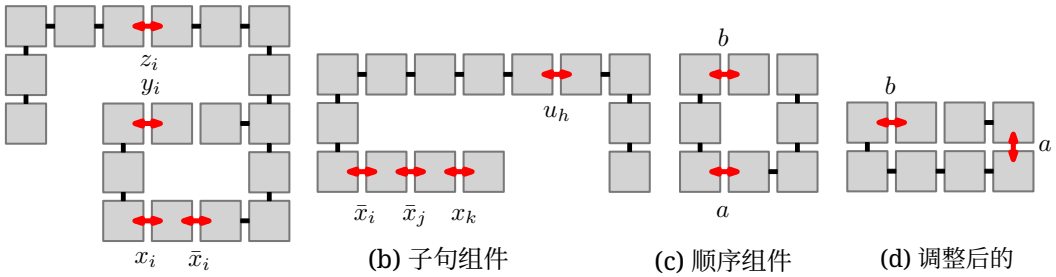
我们现在按如下方式检测可能发生的碰撞：首先，我们测试是否能在任何基本环中形成环冲突。整个过程可以在线性时间内轻松完成。如果发现了碰撞，检测就结束了。如果没有，我们必须检查节点碰撞。在第2节中，我们观察到：

▶**观察6**。对于一个含有不会导致环碰撞的环的图 S，执行任意一部分操作后，占用单元与 S 的任意生成树所对应的占用单元相同。

因此，我们计算 S 的任意一棵生成树 T，并在 T 上使用第3.1节的算法来查找节点碰撞。对于 T 中省略了 S 的边的节点对，不会发生节点碰撞。这样便将第3.1节的结果从树推广到了通常的连通图，同时保持相同的时间复杂度界。

4 连续与离散无碰撞调度

到目前为止，我们考虑的是如何检测给定输入实例是否可能发生碰撞。本节将考虑这样一个问题：能否通过选择合适的操作顺序或开始时间，使所有操作都能在不发生碰撞的情况下完成。我们将证明，即使只有水平扩张，且耦合的大小为 $O(1)$，在离散时间模型中，该问题仍是 NP完全。有趣的是，同一个问题可以在



(a) 变量组件　(b) 子句组件 用于 $x_i \vee x_j \vee \bar{x}_k$　(c) 顺序组件　(d) 调整后的顺序组件

图 5 用于证明无碰撞调度 NP-困难性的构件。(c) 中的构件用于离散情形；连续情形则需要使用 (d) 中所示的构件。

连续时间模型中用多项式时间求解。加入垂直扩展后，该问题在连续时间模型中是 NP困难。

▶**定理 7**。 即使所有操作 s 都是水平扩展，且所有耦合的规模均为 $O(1)$，离散无碰撞调度仍然是 *NP*完全问题。

证明。 为证明 NP困难性，我们将以下问题归约到 离散无碰撞调度。

LSAT（线性 SAT）。 给定一个 CNF 公式，其中每个子句恰好包含三个文字，每个子句至多与另一个子句共享一个文字，并且任意两个子句至多共享一个文字，判断该公式是否存在满足赋值。

LSAT 是 NP完全 [6]。令 ϕ 为一个包含 n 个变量 $\{x_1, \ldots, x_n\}$ 和 m 个子句的 LSAT 公式。对于公式 ϕ，我们构造一个离散无碰撞调度实例，使得当且仅当 ϕ 可满足时，该实例存在无碰撞调度。

对于每个变量 x_i，我们使用图 5(a) 所示的 变量组件 。我们执行操作 x_i 或 $\bar{x}_i$，分别将变量 x_i 设为真或假。如果尝试同时将变量 x_i 设为真和假，该组件就会导致碰撞。我们使用操作 y_i 来强制设定该变量，随后使用操作 z_i 来确保另一个操作（$\bar{x}_i$ 或 x_i）也能够执行。

对于每个子句，我们使用图 5(b) 所示的 子句组件 。该组件为每个文字都包含一个操作，并将其与相应变量组件中的逆操作耦合，即当且仅当该文字被设为假时，才会执行该操作。当且仅当三个操作全部执行时，该组件才会导致碰撞，也就是说，三个文字都被设为假；这里我们假设操作 u_h 尚不能执行。因此，发生碰撞意味着该子句不满足。将所有变量设定后，当且仅当不存在碰撞时，我们才能为 ϕ 找到一个满足赋值。

还需要两个要素：首先，我们必须确保如果 ϕ 可以满足，那么最终所有操作都能够执行，因为离散无碰撞调度要求我们执行所有操作。为此，我们在每个变量组件和子句组件中都加入了一个释放操作，分别是 z_i 和 u_h。通过扩张这些操作，我们就能获得足够的空间来扩张所有 x_i和 $\bar{x}_i$。

其次，我们必须确保在所有变量都设定之前，无法执行释放操作。为此，我们引入图 5(c) 所示的顺序组件 。当且仅当我们先执行操作 b 再执行 a 时，该组件才会导致碰撞。因此，它强制规定了 a 和 b 之间的顺序。令 $a \prec b$ 表示 a 必须在 b 之前执行。我们定义如下顺序

操作之间：

$$y_1 \prec y_2 \prec \cdots \prec y_n \prec z_1 \prec z_2 \prec \cdots \prec z_n \prec u_1 \prec u_2 \prec \ldots u_m.$$

链中的每个关系都需要一个顺序组件，总共需要 $2n+m-1$ 个。执行 y_n 时，每个变量都必须已设定为真或假。这就是对 ϕ 进行真值赋值检验的时刻。请注意，在离散时间模型中，每组耦合操作都必须完成后，下一组才能开始，因此在所有 y 完成之前，不能开始任何 z 或 u ；而这要求每个变量都已设定。

我们在各个组件中仅使用了水平扩展。此外，在 LSAT 中，每个文字最多出现两次。因此，每个操作 x_i 和 $\bar{x}_i$ 在变量组件中恰好出现一次，在子句组件中最多出现两次。每个其他操作在变量组件或子句组件中出现一次，在顺序组件中最多出现两次。因此，每个耦合的大小都是固定的。我们可以将所有组件连接成一个线性大小的单一树结构，从而生成一个连通形状 S。

最后，请注意，离散无碰撞调度属于 NP，因为我们可以在多项式时间内轻松检查任何调度是否存在碰撞。◀

我们切换到连续时间模型，并给出一个正面结果和一个负面结果，以补充离散时间模型中的 NP完全结果。

▶**定理 8.** 如果所有操作都是水平的，则无碰撞调度可以在线性时间内求解。

证明。 我们将证明，当且仅当所有操作同时执行时不会发生碰撞，才存在无碰撞调度。注意，耦合无关紧要：无论如何，所有操作都会同时执行；在连续时间模型中，即使存在耦合，这样做也是允许的。每个节点都以固定速度从初始位置移动到目的地。

注意，如果所有操作都是水平的，就没有节点能离开所在行。因此，我们可以针对每一行分别设定避免碰撞的约束。考虑形状 S 的任意一行中的任意间隙：由空网格单元构成的有界极大子行。操作可以使间隙缩小或扩大一个单位，其他操作则不会影响它。间隙的无碰撞约束规定：扩大间隙的操作数量加上间隙的初始大小，必须大于或等于缩小间隙的操作数量。当且仅当没有间隙的大小变为负数时，才不会发生碰撞。

沿着（树）形状的外侧遍历并维护一个栈，就能在线性时间内确定每个间隙两侧的节点。我们也能在线性时间内确定每个节点最终的位置，从而得到这些约束。◀

这个证明可以看作是在存在 k 个操作时，对 k 维空间中可行区域的研究。间隙约束是线性约束。如果初始形状没有碰撞，且最终形状也没有碰撞，那么这两种形状都位于可行区域中；由于可行区域是凸集，它们之间的线段也位于可行区域中。在 k 维空间中，这条线段对应于并行执行所有操作。

▶**定理 9。** 无碰撞调度是 *NP*困难。

证明。 为了证明 无碰撞调度的 NP 困难性，我们调整顺序组件，使其不仅能强制操作遵循特定顺序，还能在时间上将它们分隔开（见图 5(d)）。为此，它采用水平扩张和垂直扩展。◀

5 结论与开放问题

我们研究了与模块化可重构机器人模型相关的算法问题，可概括为“能否引发碰撞？”和“能否避免碰撞？”，同时执行规范中规定的所有操作。所有引发碰撞问题的变体都可在多项式时间内求解，但避免碰撞问题并非如此。

最引人关注的开放问题是：不通过耦合来避免碰撞的一般情形究竟是 NP困难，还是可以在多项式时间内解决。另一个值得探究的问题是，借助耦合，能否在次二次时间内实现碰撞。

参考文献

1 Nada Almalki 和 Othon Michail. 通过增长操作构造几何形状. 载于《无线网络算法学》：第18届无线网络算法学国际研讨会，ALGOSENSORS 2022，德国波茨坦，2022年9月8日至9日，论文集，第1–17页. 施普林格，2022. 2 Abdullah Almethen、Othon Michail 和 Igor Potapov. 推线大有裨益：可编程物质的高效通用集中式变换. 理论计算机科学，830：43–59，2020. 3 Greg Aloupis、Sébastien Collette、Mirela Damian、Erik D. Demaine、Dania El-Khechen、Robin Flatland、Stefan Langerman、Joseph O'Rourke、Val Pinciu、Suneeta Ramaswami 等. 晶体机器人（及遥控机器人）的现实重构. 机器人算法基础 VIII，第433–447页，2009. 4 Greg Aloupis、Sébastien Collette、Mirela Damian、Erik D. Demaine、Robin Flatland、Stefan Langerman、Joseph O'Rourke、Suneeta Ramaswami、Vera Sacristán 和 Stefanie Wuhrer. 立方体式模块化机器人的线性重构. 计算几何，42(6-7)：652–663，2009. 5 Greg Aloupis、Sébastien Collette、Erik D. Demaine、Stefan Langerman、Vera Sacristán 和 Stefanie Wuhrer. 使用 $O(\log n)$ 次并行移动重构立方体式模块化机器人. 载于《算法与计算：第19届国际研讨会，ISAAC 2008，澳大利亚黄金海岸，2008年12月15日至17日。论文集19》，第342–353页. 施普林格，2008. 6 Esther M. Arkin、Aritra Banik、Paz Carmi、Gui Citovsky、Matthew J. Katz、Joseph S. B. Mitchell 和 Marina Simakov. 彩色点的选择与覆盖. 离散应用数学，250：75–86，2018. 7 Joshua J. Daymude、Kristian Hinnenthal、Andréa W. Richa 和 Christian Scheideler. 通过可编程粒子进行计算.《移动实体分布式计算：移动与计算领域的当前研究》，第615–681页，2019. 8 Joshua J. Daymude、Andréa W. Richa 和 Christian Scheideler. 规范 amoebot 模型：算法与并发控制. 载于第35届分布式计算国际研讨会（DISC 2021）. Schloss Dagstuhl-Leibniz-Zentrum für Informatik，2021. 9 Mark de Berg、Otfried Cheong、Marc J. van Kreveld 和 Mark H. Overmars.《计算几何：算法与应用》，第3版. 施普林格，2008. URL: `https://www.worldcat.org/oclc/227584184`. 10 Erik D. Demaine、Sándor P. Fekete、Phillip Keldenich、Henk Meijer 和 Christian Scheffer. 协同运动规划：以有界伸长重构一群带标签的机器人. SIAM计算杂志，48(6)：1727–1762，2019. 11 Erik D. Demaine 和 Tomohiro Tachi. Origamizer：折叠任意多面体的实用算法. 载于第33届计算几何国际研讨会（SoCG 2017）. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik，2017. 12 Zahra Derakhshandeh、Shlomi Dolev、Robert Gmyr、Andréa W. Richa、Christian Scheideler 和 Thim Strothmann. Amoebot——一种可编程物质的新模型. 载于第26届ACM算法与体系结构并行研讨会论文集，第220–222页，2014. 13 Esther Ezra 和 Micha Sharir. 三维空间中三角形的射线投射及相关问题. SIAM计算杂志，51(4)：1065–1095，2022. 14 Michael Feldmann、Andreas Padalkin、Christian Scheideler 和 Shlomi Dolev. 通过可重构电路协调 amoebot. 计算生物学杂志，29(4)：317–343，2022. 15 Elliot Hawkes、B. An、Nadia M. Benbernou、H. Tanaka、Sangbae Kim、Erik D. Demaine、D. Rus 和 Robert J. Wood. 通过折叠实现可编程物质. 美国国家科学院院刊，107(28)：12441–12445，2010. 16 George B. Mertzios、Othon Michail、George Skretas、Paul G. Spirakis 和 Michail Theofilatos. 增长图的复杂性. 载于《无线网络算法学》：第18届无线网络算法学国际研讨会，ALGOSENSORS 2022，德国波茨坦，2022年9月8日至9日，论文集，第123–137页. 施普林格，2022.

17 Daniela Rus 和 Marsette Vona。晶体机器人：采用可压缩单元模块的自重构。*Autonomous Robots*，10:107–124，2001年。18 Jacob T. Schwartz 和 Micha Sharir。关于钢琴搬运工问题：III. 多个独立物体的运动协调：圆形物体在多边形障碍物之间移动的特殊情况。*The International Journal of Robotics Research*，2(3):46–75，1983年。19 Damien Woods、Ho-Lin Chen、Scott Goodfriend、Nadine Dabby、Erik Winfree 和 Peng Yin。多对数时间内算法形状与模式的主动自组装。载于第*4*届理论计算机科学创新会议论文集，第353–354页，2013年。