arXiv:2304.01764v1 [cs.RO] 4 Apr 2023

最小化桌面物体重排中的运行缓冲区：复杂性、快速算法与应用

KaiGao Si、Wei Feng、Baichuan Huang 和 Jingjin Yu

摘要

在使用上方抓取对桌面上的物体进行重排时，可能需要将物体临时移至某些缓冲空间。这自然引出了一个问题：要使某些类别的桌面重排问题可行，需要多少个可同时使用的存储空间，即“运行缓冲区”？在这项工作中，我们研究了有标签和无标签两种情形下的问题。在结构方面，我们观察到，最少运行缓冲区数量 MRB 可以通过从问题实例抽象得到的依赖图来求得，并证明计算 MRB 是 NP难。随后我们证明，在有标签和无标签两种情形下，即使是均匀圆柱形物体，所需的运行缓冲区数量也可能随着待重排物体数量的增加而无限增长。我们进一步证明，无标签情形下的界是紧的。在算法方面，我们开发了有效的精确算法，用于求解有标签和无标签桌面重排问题中的 MRB ，即使物体密度极高，也能扩展至一百多个物体。更重要的是，我们的算法还能计算出一系列操作，以验证所求得的 MRB ，并可用于解决物体重排任务。利用这些算法进行实证评估后发现，更贴近真实场景配置的随机有标签和无标签实例通常具有相当少的 MRB。通过真实机器人实验，我们证明，运行缓冲区抽象能够在紧凑、有界的工作空间中，为大量物体的原地重排带来最先进的解。

1 引言

在我们日常生活的几乎所有方面，无论是工作、居家还是娱乐，都需要抓取和重排物体，例如整理凌乱的桌面、饭后清理餐桌，或拼装拼图。同样，许多工业与物流应用需要反复重排大量物体，例如使用机器人在传送带上对产品进行分拣与包装。高效完成这些任务，对于提升相关利益方的竞争力至关重要。

然而，即使没有抓取这一难题，要确定物体操作顺序以优化重排任务也并非易事。为此， [1] 研究了采用上方抓取的桌面物体重排(TORO)问题。在该问题中，物体可以被拾起、移开，然后放置在不会与其他物体发生碰撞的位姿上。被拾起但无法直接放到目标位置的物体，会暂时存放在缓冲区中。例如，对于图1所示的场景，使用单个机械臂时，必须先将可口可乐或Pepsi移至缓冲区，才能完成任务。他们证明，计算使缓冲区总数最少的抓取与放置序列是NP难的，并提出了适用于物体数量为几十个以内时计算该解的方法。



图1：一个 TORO 实例 ance，其中三个汽水罐需要从左侧配置重新排列到右ιϡ配置 t 配置

在本研究中，我们考察了一个或许更实用的目标函数：在求解 TORO 实例时，将运行缓冲区数量降至最低（RB）。换言之，我们希望找到重排计划，使任意时刻存放在缓冲区中的物体数量达到的最大值最小，并假设每个物体至多被移至临时位置一次。我们将这一数量记为 MRB （运行缓冲区最小数量）。 MRB 目标之所以重要，是因为如果求解 TORO 实例所需的 MRB 超过实际可用的缓冲区存储空间，那么该实例便不可行。实际中的缓冲区存储空间有限。此外，该目标也有助于计算高质量解来完成 TORO 任务。因此，我们提出的结构性结果和算法不仅可用于计算可行且高质量的重排计划，也可作为重要的验证工具，例如验证某种重排场景是否能够完成其设计目标所涵盖的大多数任务。

围绕以 MRB 目标研究 TORO，本文作出以下贡献：

- 我们提出并开展了对 TORO 任务的初步研究，重点是最小化运行缓冲区的数量（MRB）。该 MRB 目标是最小化缓冲区总数的一种自然替代方案 [1], 而这两个目标都可作为衡量高质量桌面重排计划的代理指标。

- 在结构方面，就计算复杂性而言，我们证明，在任意依赖图上计算 MRB 是 NP 难的；这些依赖图编码了 TORO 实例的组合信息。随后我们进一步证明，计算 MRB 的 NP 难性也适用于实际的 TORO 实例。

- 继续对 MRB 目标进行结构分析，我们证明，对于一个包含 n个物体且所有物体均为均匀圆柱体的 TORO 实例，即使所有物体均为无标签，其 MRB 也有下界 $\Omega(\sqrt{n})$。这意味着在有标签设定下同样如此。随后，我们为无标签情形给出了与之匹配的算法上界 $O(\sqrt{n})$。

- 在算法方面，我们开发了多种有效方法，以求解各种优化 MRB的 TORO 任务，包括用于有标签设定的动态规划方法、用于无标签情形的基于优先队列的算法，以及一种新颖的深度优先搜索动态规划程序。对于这两种设定，该程序都能轻松扩展到包含一百多个物体的实例。此外，我们还提供了计算满足 MRB约束且总缓冲区数最少的规划方案的方法。这些算法不仅能给出最优缓冲区数，还能提供一个可证明该最优解的重排计划。

- 通过大量数值评估和实验验证，我们确认 MRB 作为一种恰当的指标/代理指标，在计算 TORO 任务的高质量解时具有优化效果。具体而言，基于 MRB的方法在许多极具挑战性的 TORO设置中，显著优于此前最先进的方法。

本文在会议版本的基础上进行了扩展 [2], ，主要增加了以下两项重要内容：（1）补充了许多理论和算法方面的结果；对于已有定理和新增定理，均提供了完整且扩展的证明；（2）新增了一项应用研究，利用 MRB 在有界工作空间内执行桌面原位重排，并进行了真实机器人实验。这部分新增内容部分基于 [3] ，同时还进行了新的改进实验（例如，图 2）。

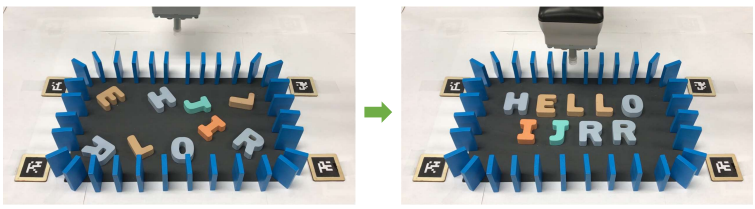


图 2：一个桌面原位重排实例，其中字母形状物体位于有界工作空间内。蓝色多米诺骨牌充当缓冲边界，并且在执行过程中不会掉落。

论文结构。本文其余部分安排如下。第2节概述相关文献。第3节介绍 MRB 所关注的重排问题，并讨论相关的依赖图结构。第4节建立一些基本的结构性质，并说明计算 MRB 在计算上难以处理。随后，我们在第MRB 节针对特定情形给出下界和上界，并在第5 节介绍所提出的算法解。第6节将介绍7节中，我们展示 MRB 算法在允许使用内部运行缓冲区的扩展场景中的应用。第8节对仿真进行评估。第9节搭建硬件平台，以执行由所提出的算法计算出的重排计划。最后，我们在第10节进行讨论并总结。

2 相关工作

作为一项实用性很高的能力，在有界工作空间内操作物体已得到广泛研究，相关工作致力于感知／场景理解 [4, 5, 6, 7]，任务／重排规划 [8, 9, 10, 11,12, 1, 13]，操作 [14, 15, 16, 17, 18, 19, 20, 21], ，以及集成式整体方法 [22, 23, 24,25, 26]。由于重排问题往往包含多机器人运动规划问题，重排也继承了 PSPACE 难度 [27]。即使没有复杂的几何约束，这些问题仍然是 NP难 [28]。考虑重排计划质量，例如尽量减少抓取与放置操作的次数或末端执行器行程，在计算上同样难以处理 [1]。

对于主要采用抓取操作的重排任务，可移动障碍物间导航 [8, 29] 的算法研究提出了回溯搜索方法，能够有效处理单调实例，即限制机器人每个障碍物最多移动一次的实例。通过谨慎调用单调求解器，也可以解决困难的非单调情形 [11, 30]。 [1] 将桌面重排问题与旅行商问题 [31] 和反馈顶点集问题 [32]，联系起来，而这两个问题均为 NP 难问题。尽管如此，整数规划模型仍能快速计算实际规模（∼ 20 个物体）问题的高质量解。研究主要聚焦于无标签情形，并针对圆盘物体给出了抓取与放置操作次数的界 [33]。在 [13]，中，研究者开发了一种完整算法，该算法能够对物体取回进行推理，并根据需要重排其他物体；后续研究 [34] 则考虑了计划最优性和传感器遮挡。尽管大多数问题的目标都侧重于运动次数，但 [35] 旨在最小化圆盘在工作空间内移动时执行重排任务所需的空间。

非抓取式重排也得到了广泛研究[36, 37], ，早期研究重点是物体分离 [38, 39, 40]。在这一问题中，机器人需要通过非抓取式操作（例如推和戳）将目标物体与周围障碍物分开，以腾出空间执行抓取操作。在 [37]中，研究人员采用迭代搜索来完成多种重排任务，包括将形状相同的立方体分离、分开和分类。 [41] 将蒙特卡洛树搜索与深度策略网络相结合，在有界工作空间内将多个物体分成连贯簇，并支持非凸物体。最近， [42], 提出了一种双层规划器，通过结合（非抓取式）推和（抓取式）俯抓，对大量物体进行分类。研究人员还探索了非抓取式操作与抓取操作之间的协同作用，以使用尽可能少的推动和抓取操作完成杂乱物体清除任务 [43, 44]以及更具挑战性的物体取回任务 [45, 46] 。

在结构层面，我们研究的一个核心对象是依赖图结构。类似的依赖结构最初被引入多机器人路径规划问题，用于处理智能体之间的路径冲突 [47, 48]。随后，这种结构被用于推理并解决具有挑战性的重排问题 [11, 49, 50, 51, 52]。由重排实例诱导的完整带标签依赖图，首次在 [1]中被提出并研究。本研究引入无标签依赖图。我们观察到，在有标签设定中，通过依赖图，运行缓冲区问题自然地与图布局问题相关联 [53, 54, 55, 56, 57, 58], ，后者旨在寻找图顶点的最优线性排序。图布局问题有大量重要应用，包括超大规模集成电路设计、调度 [59], 等。在无标签情形下，对于具有圆形底座的均匀物体，依赖图是平面图。可以通过使用顶点分隔集对依赖图进行划分来解决重排问题 [60, 61, 62, 63]。有关这些主题的综述，请参见 [53]。

3 预备知识

对于物体几何形状相似的 TORO 任务，根据物体是否可区分（即是否为有标签或无标签），存在两种自然的实际设定。我们介绍这两种不同设定下的外部缓冲区表述，并讨论两种设定中重要的依赖图结构。

3.1 带外部缓冲区的标记 TORO

考虑一个有界工作空间 $\mathcal{W} \subset \mathbb{R}^2$，其中放置了一组 n 物体 $\mathcal{O} = \{o_1, \ldots, o_n\}$。假设所有物体都是高度相同的广义圆柱体。这些物体的一个可行排列是一个位姿集合 $\mathcal{A} = \{x_1, \ldots, x_n\}, x_i \in SE(2)$，其中任意两个物体均不发生碰撞。设 $\mathcal{A}_1 = \{x_1^s, \ldots, x_n^s\}$和 $\mathcal{A}_2 = \{x_1^g, \ldots, x_n^g\}$ 为两个可行排列，一个桌面物体重排问题 [1] 旨在寻找一个规划方案，通过抓取放置操作将物体从 $\mathcal{A}_1$ 移动到 $\mathcal{A}_2$（示例见图 3(a)，其中包含 7 个均匀圆柱体）。在每个抓取放置操作中，机器人用机械臂抓取一个物体，将其提升至所有其他物体之上，移动并降落到一个新的位姿 $p \in SE(2)$，使该物体不会与其他物体发生碰撞，然后将其释放。抓取放置操作可形式化表示为一个三元组 $a = (i, x', x'')$，表示物体 o_i 从位姿 x' 移动到位姿 x''。完整的重排计划$P = (a_1, a_2, \ldots)$随后是一个抓取放置操作的有序序列。

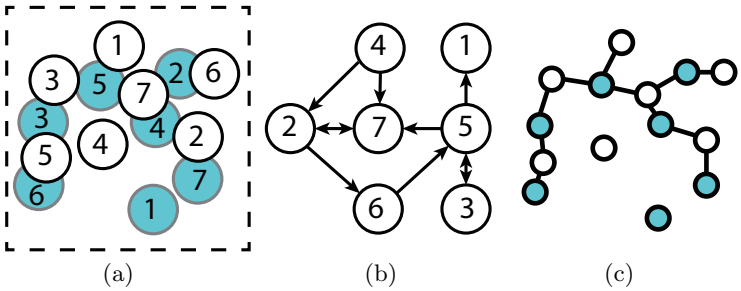


图 3：一个包含 7 个物体、由均匀圆柱体构成的有标签实例；我们将以此实例作为贯穿全文的示例。(a) 未着色的圆盘（作为圆柱体的投影）表示起始布局 $\mathcal{A}_1$，着色的圆盘表示目标布局 $\mathcal{A}_2$。(b) 对应的带标签依赖图。(c) 对应的无标签依赖图，该图是二分平面图。

根据 $\mathcal{A}_1$ 和 $\mathcal{A}_2$，有时无法通过一次抓取并放置操作直接将物体 o_i 从 x_i^s 移至 x_i^g，因为x_i^g 可能被其他物体占据。这会在物体之间形成依赖关系。如果处于位姿 x_i^g 的物体 o_i 与处于位姿 x_j^s的物体 o_j 相交，我们就说 o_i 依赖 o_j。这意味着必须先移动物体 o_j，然后才能将 o_i 放置到其目标位姿 x_i^g。

可能存在循环依赖。例如，对于图 3(a) 中给出的实例，物体 3 和物体 5 彼此依赖。在这种情况下，必须将某些物体临时移至中间位姿，才能解决重排问题。与 [1], 本文的大多数内容一样，我们假设使用 工作空间 之外的外部缓冲区来存放中间位姿，这样一来，若中间位姿需放置在 $\mathcal{W}$内，就可以避免耗时的几何计算。在执行重排计划的过程中，可能有多个物体存放在缓冲区位置。我们将当前正在使用的缓冲区称为 运行缓冲区 (RB)。引入缓冲区后，抓取放置操作分为三种：1）抓取处于起始位姿的物体并将其放置到缓冲区；2）抓取处于起始位姿的物体并将其放置到目标位姿；3）从缓冲区抓取物体并将其放置到目标位姿。请注意，缓冲位姿并不重要。显然，最好能够用最少数量的运行缓冲区解决重排问题，由此引出带标签的运行缓冲区最小化问题。

问题 1（带标签的运行缓冲区最小化问题（LRBM））。 给定可行排列 $\mathcal{A}_1$ 和 $\mathcal{A}_2$，求一个重排计划 P，使得执行规划方案的任意时刻所使用的运行缓冲区数量最大值最小。

在 LRMB 实例中，由 $\mathcal{A}_1$ 和 $\mathcal{A}_2$ 诱导的所有依赖关系集合可以用有向图 $G^\ell = (V, A)$表示，其中每个 $v_i \in V$对应一个物体 o_i，若物体 o_i依赖于物体 o_j，则存在一条弧 $v_i \to v_j$for$1 \le i, j \le n, i \ne j$。我们将 G^ℓ 称为带标签的依赖图。图 3(a)对应的带标签的依赖图如图 3(b)所示。根据依赖图 G^ℓ，我们可以立即识别出图中的多个循环依赖，例如物体 3 和 5 之间的循环依赖，或物体 7、2、6 和 5 之间的循环依赖。这些环构成了 G^ℓ的强连通分量，可以通过 [64]有效计算。由于将物体移至外部缓冲区不会产生额外的依赖关系，因此我们有

命题1。 G^ℓ 完整地捕获了使用外部缓冲区将物体从 $\mathcal{A}_1$ 移动到 $\mathcal{A}_2$以解决桌面重排问题所需的信息。

我们使用两个示例来说明 TORO、其依赖图与基于外部缓冲区的解之间的关系。首先，对于图 1中的示例，对应的起始/目标配置如图 4[左上]所示。带标签的依赖图如图 4[右上]所示。由于可口可乐和Pepsi之间存在循环依赖，其中一个必须暂时移动到缓冲区中。假设将Pepsi移动到缓冲区。这会改变配置和依赖图，如图 4第二行所示。随后即可轻松解决该 TORO 实例。完整的解序列为
$\langle Pepsi \to b, Coke \to g, Pepsi \to g, Fanta \to g \rangle$，其中 b指缓冲区，而 g指相应物体的目标位置。

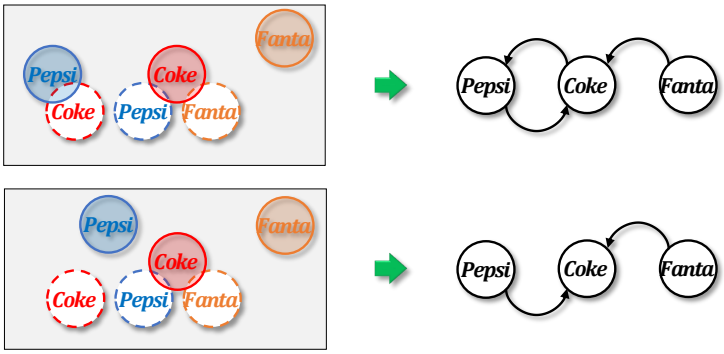


图 4：图中所示设置的两种配置。1以及相应的依赖图。

对于图 3[a], 中的示例，其带标签的依赖图（图 3[b] ）显示了两个环（$2 \leftrightarrow 7$ 和$3 \leftrightarrow 5$）。可以推导出一种解决此问题的规划序列，将物体 2 和 3 移至缓冲区： $\langle 1 \to g, 2 \to b,\ 3 \to b,\ 7 \to g,\ 4 \to g,\ 5 \to g,\ 6 \to g,\ 2 \to g,\ 3 \to g \rangle$。

3.2 无标签 TORO 与外部缓冲区

在无标签情形中，物体可以互换。也就是说，哪个物体前往哪个目标并不重要。例如，在图 3中，物体 5 可以移至物体 6 对应的目标。我们将此版本称为无标签运行缓冲区最小化 （URBM）问题，直观上它更容易求解。无标签问题的规划方案可以像有标签设定那样表示；我们继续使用标签，以便清晰地表示规划方案，但不要求起始位姿和目标位姿的标签相匹配。

对于无标签情形，依赖结构依然存在，但呈现形式不同。它现在是一个无向二分图。也就是说， $G^u = (V_1 \cup V_2, E)$，其中每个 $v \in V_1$ （分别对应 $v \in V_2$）对应一个起始（分别对应目标）位姿 $p \in \mathcal{A}_1$ （分别对应 $p \in \mathcal{A}_2$）。我们将表示起始位姿和目标位姿的顶点分别称为起始顶点和目标顶点。如果相应位姿处的物体发生重叠，则 $v_1 \in V_1$ 与 $v_2 \in V_2$ 之间存在一条边。图3（a）对应的无标签依赖图如图3（c）所示。

实际上，许多 TORO 实例中的物体，其占地轮廓为全等正多边形（例如正方形）或圆盘。在这种情况下，我们还能对由此得到的无标签依赖图得出一些额外结论（以下两个命题的证明见附录）。

命题2。 对于占地轮廓为全等正多边形的无标签 *TORO* 物体，其无标签依赖图的最大度数不超过 *19*。

命题 3。 对于物体占地轮廓为全等圆盘的无标签 *TORO*，其依赖图是一个最大度数为 *5* 的平面二分图。

对于 LRBM 或 URBM，所需的最小运行缓冲区数量记为 MRB，可根据相应的依赖图计算得出。

4 结构分析与 NP困难性

在桌面重排问题中引入运行缓冲区，会产生独特而有趣的结构。我们重点介绍 LRBM 的一些重要结构性质，包括与最小化缓冲区总数的比较 [1], LRBM 的解 与其依赖图的线性排列 [65] 或线性排序[66] 之间的关系，以及计算 MRB TORO 的难度。

正如本节将要证明的，计算 TORO 的 MRB 解与在相关依赖图中寻找某种最优顶点线性排序密切相关。由于寻找最优线性排序是困难的，因此计算 MRB 解也同样困难。这也限制了人们能够为计算 TORO任务的 MRB 解所获得的算法解。

4.1 运行缓冲区与总缓冲区

在求解 TORO 时，运行缓冲区与缓冲区总数相关但并不相同，正如[1]中所研究的那样。研究表明，求解 TORO 所需的最小总缓冲区数与底层依赖图的最小反馈顶点集 （FVS）的大小相同。FVS 是一个顶点集，移除该顶点集后，图将变为无环图。具有无环依赖图的 TORO 无需使用任何缓冲区即可求解，因为任意物体对之间不存在循环依赖。我们将最小 FVS 的大小记为 MFVS。

作为一个带标记的 TORO 问题，图 1 中的示例具有 $\mathrm{MFVS} = \mathrm{MRB} = 1$。对于图 3（a）（b）中的示例，$\mathrm{MFVS} = 2$（FVS 集为 $\{2, 3\}$），并且 $\mathrm{MRB} = 2$。

为了说明 MRB 与 MFVS之间的极端差异，考虑一个带标记的依赖图，该图由 n 个二元环副本组成，例如图 5所示。该实例的 MFVS 为 n，因为必须从每个 n 环中移除一个顶点，才能使图变为无环图。另一方面，MRB 仅为 1，因为每个循环依赖都可以独立于其他环，使用一个外部缓冲区加以解决。因此，尽管所使用的总缓冲区数与全局解最优性关系更为密切， MRB 却更能体现 可行性。了解 MRB 可以帮助我们判断，对于某一类重排问题，一定数量的外部缓冲区是否足以完成求解。这对于实际应用至关重要，因为外部缓冲区的数量通常受到限制，只能保持为一个较小的常数。例如，在处理日常重排任务时，如整理物品或从冰箱中取出物品，人可能会尝试暂时拿住多个物品，但有时会很不方便。显然，以这种方式能够同时拿住的物品数量是有限的。

我们举例说明， MRB 和 MFVS 并不总能同时优化，也就是说，它们构成一个帕累托前沿。对于图 6 中的设置， MFVS $\{7, 9, 10\}$ 的大小为 3——不难验证，删除 $\{7, 9, 10\}$ 后，依赖图便是无环的。使用我们的算法，可以计算出 MRB 为 2，其见证序列为 1、10、8、4、5、3、6、7、2、9，对应的规划方案为 $\langle 1 \to g, 10 \to b$、$8 \to g$、$4 \to b$、$5 \to g$、$3 \to g$、$10 \to g$、$6 \to b$、$7 \to g$、$6 \to g$、$2 \to b$、$9 \to g$、$4 \to g$、$2 \to g\rangle$，其中 b 表示一个缓冲区， g 表示相应物体的目标。当 4 和 6 被移入缓冲区时，运行缓冲区达到最大值 2。不过，在这种情况下，使用的缓冲区总数为 4：2、4、6 和 10 都被移入缓冲区。事实证明，这是该示例中能达到的最佳结果；也就是说，如果我们限定解满足 $\mathrm{MRB} = 2$，那么必须使用的缓冲区总数至少为 4，而不是 MFVS大小 3。

我们注意到，当解空间被限制为MRB 解时， MFVS 很少会更大；例如，对于均匀圆柱体，在首先最小化运行缓冲区后，所需缓冲区的总数几乎总是与 MFVS 大小相同。

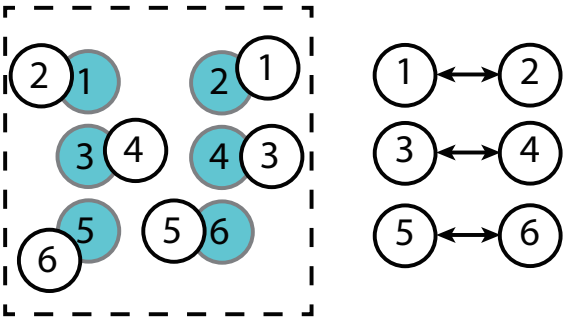


图 5：一个实例，其带标签的依赖图由 3 个二元环副本构成。 MFVS 为 3。另一方面，该问题的 MRB 仅为 1。此例可以扩展，使 MFVS 任意大，同时 MRB 仍为 1。

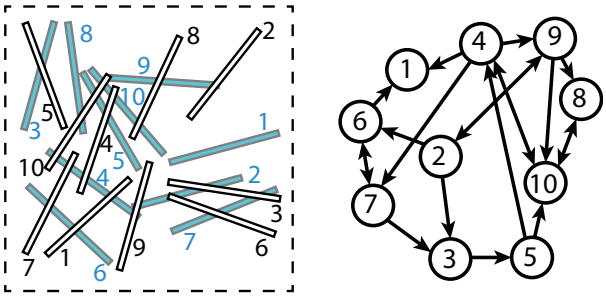


图 6：一个由均匀细长长方体组成的 LRBM 实例（左）及其带标签的依赖图。在运行缓冲区数量最小化时，所需的缓冲区总数大于 MFVS 的大小。

4.2 运行缓冲区与线性顶点排序

给定顶点集为 V的图， V的线性排序 是一个双射函数 φ：$\{1,\ldots,|V|\} \to V$。对于 LRBM 问题的带标签依赖图 $G^{\ell}(V,\ A)$，以及 V的线性排序 φ，我们可以通过依次拾取与顶点 $\varphi(1)$ $\varphi(2)$ ...相对应的物体，将 φ 转化为该 LRBM 问题的规划方案 P。对于每个被拾取的物体，如果它没有更多依赖关系，就将其移至目标位姿；否则，将其存放在缓冲区中。已在缓冲区中的物体将在最早可能的时机移至其目标位姿。

例如，对于图 中的依赖图，给定线性排序 5，6，3，4，2，7（图 3（b）），首先可以将 1 直接移动到其目标位置。接着，由于 5 依赖于 3 和 7（但不再依赖于 1），因此将 5 移至缓冲区。然后，由于 5 现在位于缓冲位置，可以将 6 直接移动到其目标位置。类似地，接下来可以将 3 移至其目标位置。之后，必须将 4 和 2 移至缓冲区，此后便可将 7 直接移动到其目标位置。最后，可以将 2，4 和 5 从缓冲区分别移至各自的目标位置。这样，运行缓冲区大小最大为 3。这并非最优；一个最优序列是 ，6，2，7，4，3，1，其中$\mathrm{MRB} = 2$。这两个序列均示于图 7。

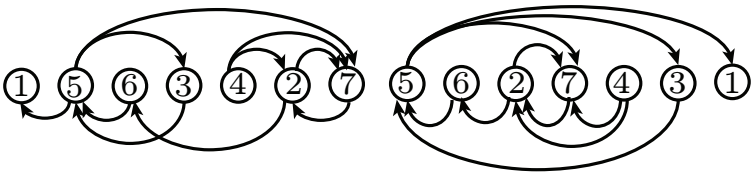


图 7：图 3(b) 中带标签的依赖图的两种顶点线性排序（即，它们是同一图的不同表示）。右侧的排序使 MRB 最小。

讨论表明，我们可以有效地将运行缓冲区的数量视为一个函数 a

依赖图 G^{ℓ}和线性排序 φ。因此，我们将 $\mathrm{RB}(G^{\ell},\varphi)$ 定义为按照 φ 给定的顺序重新排列 G^{ℓ} 所需的运行缓冲区数量。显然， $\mathrm{MRB}(G^{\ell})=\min_{\varphi}\mathrm{RB}(G^{\ell},\varphi)$。

4.3 计算 MRB 的困难性

由于计算 MFVS 是 NP难的， [1], 因此可以预期，对于一个带标签的依赖图（可以是任意有向图），计算 MRB 同样很困难。我们通过考察 MRB 与顶点分离问题（ VSP ）之间有趣的关系来证明这一点。如定理3.1中所述，该问题等价于路径宽度、门矩阵布局和搜索数问题；该定理 [53], 源于一系列研究 [67, 68, 69]。除非 $P=NP$，否则这些问题中的任何一个都不存在绝对近似算法 [58]。首先，我们介绍顶点分离问题。直观地说，给定一个无向图 $G=(V,E)$，VSP 旨在为 φ 中的 V 顶点确定一个线性排序，使得对于排序为 i的顶点，将排序中不晚于 i 且与排序中晚于 i的顶点通过边相连的顶点数量最小化。

顶点分离（顶点分离问题）
实例：图 $G(V,E)$ 和一个整数 K。**问题**：是否存在双射函数 $\varphi:\{1,\ldots,n\}\to V$，使得对于任意整数 $1\le i\le n$, $|\{u\in V\ \exists(u,v)\in E\ and\ \varphi(u)\le i<\varphi(v)\}|\le K$?

例如，在图8(a)中，给定线性排序后，在第二个顶点处，第一个和第二个顶点都存在边与垂直分隔器交叉，因此交叉数为2。给定图 G和线性排序 φ，我们定义 $\mathrm{VS}(G,\varphi):=\max_i|\{u\in V\mid\exists(u,v)\in E\ and\ \varphi(u)\le i<\varphi(v)\}|$，顶点分离问题（VSP）寻求$\varphi$ 使 $\mathrm{VS}(G,\varphi)$最小化。设 $\mathrm{MinVS}(G)$为图 G的顶点分离数，即使顶点分离问题实例的答案为“是”的最小 K，则 $\mathrm{MinVS}(G)=\min_{\varphi}\mathrm{VS}(G,\varphi)$。现在，给定一个无向图 G 和一个带标签的依赖图 G^{ℓ}，后者由 G 通过将 G 的每条边替换为两条方向相反的有向边得到。我们观察到 $\mathrm{VS}(G,\varphi)$与 $\mathrm{RB}(G^{\ell},\varphi)$之间存在明显相似之处，以下引理对此进行了刻画。

引理1。 $\mathrm{VS}(G,\varphi)\le\mathrm{RB}(G^{\ell},\varphi)\le\mathrm{VS}(G,\varphi)+1$。

证明。 固定线性排序 φ后，显然有 $\mathrm{VS}(G,\varphi)\le\mathrm{RB}(G^{\ell},\varphi)$，因为对于 G，分隔器左侧且有边跨越该分隔器的顶点对应于必须存放在缓冲区位置的物体。例如，在图 8(a) 中，经过从左侧数第二个顶点后，第一个和第二个顶点都有边跨越垂直的“分隔器”。在图 8(b) 所示的相应依赖图中，与这两个顶点对应的物体都必须移至外部缓冲区。另一方面，我们有 $\mathrm{RB}(G^{\ell},\varphi)\le\mathrm{VS}(G,\varphi)+1$，因为当我们沿线性排序经过一个顶点时，相应的物体可能需要暂时移至缓冲位置。例如，在图 8(a) 中经过从左侧数第三个顶点后，顶点分隔集从 2 降至 1；但在处理图 8(b) 中相应的依赖图时，必须先将与从左侧数第三个顶点对应的物体移至缓冲区，然后才能将存放在缓冲区中的第一个和第二个物体放置到各自的目标位置。

□

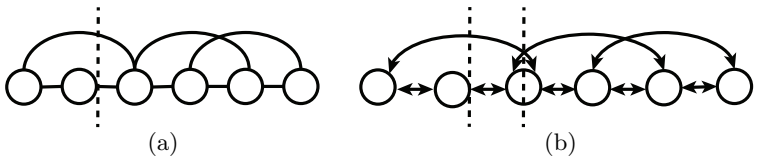


图 8: （a）一个无向图及其顶点的线性排序。（b）与之对应的带标签依赖图，顶点排序相同。

定理 1。 对于带标签的依赖图，即使采用绝对近似，计算 MRB 仍是 *NP*难问题。

证明。 给定无向图 G，我们将 VSP 的常数因子近似归约为依赖图 G^ℓ 上 MRB 的常数因子近似，其中 G^ℓ 由 G 按照前文所述构造，并将 G 中的每条边替换为双向依赖关系。

除非 $P = NP$，否则 VSP 不存在多项式时间的绝对近似。因此，若 $\mathrm{MRB}(G^\ell, \varphi)$ 可在多项式时间内近似到 α，也就是说，对于图 G，我们可以在多项式时间内找到一个 φ^*，使得 $\mathrm{RB}(G^\ell, \varphi^*) \leq \mathrm{MRB}(G^\ell) + \alpha$，那么我们就有 $\mathrm{VS}(G, \varphi^*) \leq \mathrm{RB}(G^\ell, \varphi^*) \leq \alpha + \mathrm{MRB}(G^\ell) \leq \textsc{MinVS}(G) + \alpha + 1$，这表明顶点分离可以有绝对近似，从而推出 $P = NP$。 □

经过一些额外的努力，我们可以证明，计算某些 TORO实例的 MRB 解是 NP难的。

定理 2. 计算 MRB 对于 *TORO* 是 *NP*难。

证明。 根据 [70], 顶点分离问题 对于最大度数为三的平面图仍然是 NP完全问题。根据引理 1和定理 1及相应的证明，如果我们能够证明可以将任意最大度数为三的平面图转换为相应的 TORO 实例，那么问题便得以解决。也就是说，我们只需证明：若图 8(a) 是一个最大度数为三的平面图，则可以构造一个 TORO 实例，使其依赖图为图 8(b)。

接下来，我们将证明一个更强的结论：我们不只证明上述结论适用于最大度为三的平面图，还将证明它适用于所有平面图。根据 [71], , 所有平面图都可以仅用直线边绘制在平面上，且没有边交叉。给定一个任意的平面图及其在平面上的一个无边交叉直线边嵌入，我们将展示如何把该嵌入转换为对应的 TORO 实例。

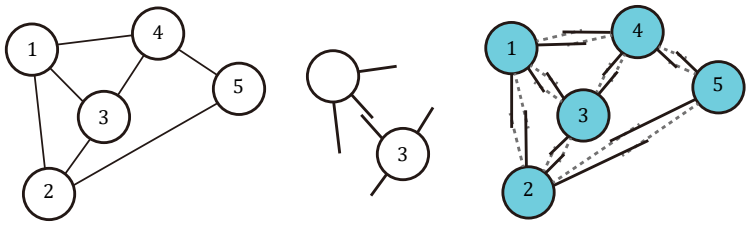


图 9: （a）一个包含五个顶点的无向平面图；（b）其中两个顶点对应的“对象构件”；（c）相应的 TORO 实例。

构造过程通过图 9 进行说明，其中图 9(a) 展示了平面图的直线边非交叉嵌入。我们根据该嵌入的几何结构，按照以下方式构造 对象构件 。对于图中的每个节点，我们取该节点以及从该节点引出的每条边略多于一半的部分。例如，对于节点 1 和 3，相应的物体如图 9(b) 所示。现在，我们构造一个 TORO 实例，其依赖图对应于将图 9(a) 中的边替换为双向边。为此，我们按照图9(a) 中所示的位置放置每个对象构件。对于目标构型，我们将每个物体围绕相应节点的中心逆时针 旋转一个小角度 ε 。对于起始构型，我们执行类似的旋转，但方向为 顺时针 。对于图 9(a)，这一过程得到图9(c)。不难验证，该构造将平面图中的每条边 (i, j) 转换为两个物体之间的相互依赖关系 i 和 j。例如，图 9(c) 中的对象构件 1 和 3 之间存在循环依赖。

□

5 MRB 的下界和上界

在将 MRB 与顶点线性排序联系起来并证明其计算不可行性之后，我们继续确定 MRB的定量界，即 LRBM 和 URBM 所需的最低 MRB 是多少，以及我们最多能将 MRB降低到什么程度?

5.1 MRB 的内在下界

当不限制物体几何形状时， MRB 可以轻易达到一个 $n-1$ 物体n 实例所可能达到的最大值，即使在URBM 情况下也是如此。这种情况的一个示例见图10，其中 $n=6$个细长长方体在 $\mathcal{A}_1$中水平排列，一个位于另一个上方。这些长方体在 $\mathcal{A}_2$中垂直对齐。于是，每一对起始位姿和目标位姿都会诱导出一个（唯一的）依赖关系。显然，这在 LRBM 情况下产生一个双向K_6 带标签的依赖图，而在 URBM 情况下产生一个 $K_{6,\ 6}$ 无标签依赖图。在这两种情况下，都必须先将 $n-1=5$ 个物体移入缓冲区，才能解决该问题。这个示例可以直接推广到任意数量的物体。

命题4. MRB 下界为 $n-1$，适用于 n 个物体，对 *LRBM* 和 *URBM* 均成立；即使对于均匀凸物体，这也是可能达到的最大值。

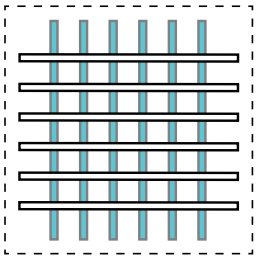


图10：一个包含6个长方体的实例，其中水平和垂直集合分别表示起始位姿和目标位姿。在有标签设定中，可以为这些物体指定任意标签。

关于 MRB为 $\Omega(n)$ 的下界虽不理想，但这似乎取决于物体较为细长；日常物体通常并非如此。因此，一个具有很高实际价值的问题是：当物体的占地轮廓“较为饱满”时会怎样？接下来，我们证明，对于均匀圆柱体，下界降至 $\Omega(\sqrt{n})$，而均匀圆柱体可以近似许多现实世界物体/产品。此外，我们证明该下界对于 URBM 是紧的（见第5.2节）。

为方便起见，假设 n 是一个完全平方数，即 $n=m^2$，其中 m为整数。为证明 $\Omega(\sqrt{n})$ 下界，我们使用一种类似网格的无标签依赖图，并称之为依赖网格，其中 $\mathcal{A}_1$和 $\mathcal{A}_2$ 具有相似的模式， $\mathcal{A}_2$ 相对于 $\mathcal{A}_1$ 向左（或向右/上方/下方）偏移一个网格边的长度。图11展示了这种布局的一部分。我们用11表示 $\mathcal{D}(w,h)$ 个列、w 列和 h 行的依赖网格。

引理2。 给定一个具有 $n=m^2$ 个物体且其依赖图为 $\mathcal{D}(m,2m)$ 的 *URBM* 实例，其 MRB的下界为 $\Omega(m)=\Omega(\sqrt{n})$

由于该证明与论文主要贡献的呈现关系有限，因此我们仅阐述其核心思想，并请读者参阅附录以获取完整证明。我们证明，对于具有此类依赖图的某些包含 n 个物体的无标签TORO实例，为了同时解决该TORO实例，至少有 $\Omega(\sqrt{n})$ 个物体必须被移动到缓冲区中。

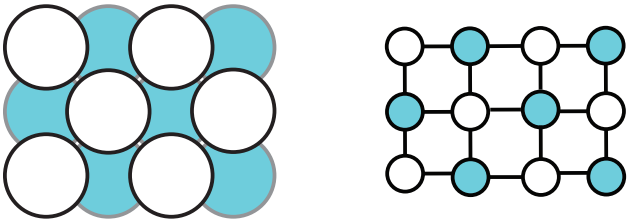


图11：一个 URBM实例（左）及其无标签依赖图（右），一个 4×3 依赖网格。未着色（分别为着色）的圆盘/顶点表示起始位姿（分别为目标位姿）。

由于 URBM 实例所需的运行缓冲区数量始终小于或等于 MRB 与相同物体和目标位置对应的 LRBM 实例，因此引理 2 的结论可直接应用于LRBM。因此，我们有

定理 3. 对于具有 n 均匀圆柱体的 *URBM* 和 *LRBM*， MRB 的下界为$\Omega(\sqrt{n})$.

对于均匀圆柱体，尽管 URBM 的下界是紧的（如第 5.2 节所示），但我们不知道 LRBM 的下界是否紧；我们猜想，$\Omega(\sqrt{n})$ 并非 LRBM 的紧下界。事实上，当均匀圆柱体仅沿一个环排列时，即可达到 $\Omega(\sqrt{n})$ 下界，图 12给出了示意图。对于一般构造，对每个物体 o_i，令 o_i 依赖于 $o_{(i-1 \text{ 模 } n)}$ 和 $o_{(i+\sqrt{n} \text{ 模 } n)}$，其中 n 是实例中的物体数量。根据带标签的依赖图，我们可以构造实际的 LRBM 实例，使其起始与目标布局均构成一个环。可以证明，当 $n/2$ 个物体处于目标位姿时，$\Omega(\sqrt{n})$ 个物体位于缓冲区。我们省略证明，其思路与引理 2的证明类似。

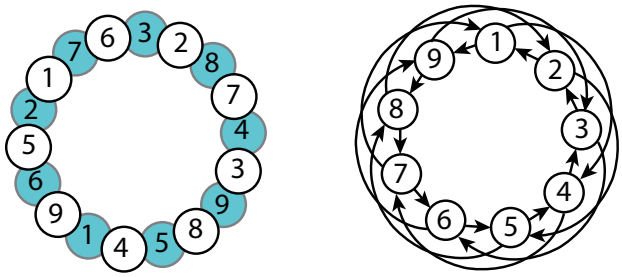


图12：一个包含9个物体的LRBM示例，其产生$\Omega(\sqrt{n})$ MRB （左），以及对应的依赖图（右）。

5.2 URBM的MRB上界

现在我们证明，无论如何重排 n 均匀圆柱体，对应的URBM实例都存在一个使用 $\text{MRB} = O(\sqrt{n})$的解。URBM的下界和上界一致；因此，$O(\sqrt{n})$界是紧的。

为证明上界，我们针对该设置提出一种基于 G^u顶点分隔集的 $O(n\log(n))$时间算法 SepPlan 。 SepPlan 计算一个待从依赖图中移除的目标顶点序列。给定待移除的目标顶点序列，每个时刻的运行缓冲区大小等于$\max(0, \|N(g, G^u)\| - \|g\|)$，其中 g是该时刻已移除目标顶点的集合， $N(g, G^u)$是 g 在 G^u中的邻居集合。我们证明， SepPlan 可以找到一个使用 $O(\sqrt{n})$个运行缓冲区的重排计划。

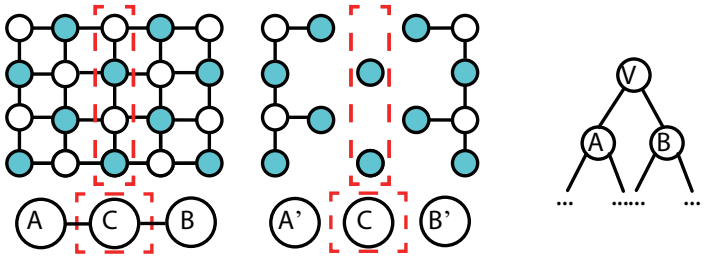
算法 1： SepPlan

输入： $G^u(V, E)$：无标签依赖图
输出： π：目标序列

$\pi,\ V,\ E \leftarrow$ RemovalTrivialGoals($G^u(V,\ E)$)
如果 V *is* $\emptyset$ **，则返回** π ;
$A, B, C \leftarrow$ 分隔器($G^u(V, E)$)
$\pi \leftarrow \pi + g(C)$
$A' \leftarrow A - N(g(C)\ G^u(V, E))$
$B' \leftarrow B - N(g(C)\ G^u(V, E))$
$\pi_{A'} \leftarrow$ SepPlan $(G^u(A', E(A')))$
$\pi_{B'} \leftarrow$ SepPlan $(G^u(B',\ E(B')))$
如果 $\delta(A') \geq \delta(B')$ **那么** $\pi \leftarrow \pi + \pi_{A'} + \pi_{B'}$;
否则 $\pi \leftarrow \pi + \pi_{B'} + \pi_{A'}$;
返回 π

该算法见算法 1。 SepPlan 使用图 $G^u(V, E)$，它是由顶点集 G^u 诱导的 V的一个子图。首先，可以在不使用缓冲区的情况下，移除孤立目标顶点或在 $G^u(V, E)$ 中只有一个依赖关系的顶点（第 1 行）。之后， V可以划分为三个

不相交的子集 A、B 和 C [60]（第 3 行），使得在 A 与 B、$|A|$、$|B| \le 2|V|/3$ 和 $|C| \le 2\sqrt{2|V|}$ 中的顶点之间不存在边（图 13(a)）。对于 C 中的起始顶点以及 C 中目标顶点的邻居，我们将它们从 G^u 中移除。由于每个目标顶点最多有 5 个邻居，因此在此操作中，最多有 $10\sqrt{2|V|}$ 个物体被移至缓冲区。之后，我们移除 C 中的目标顶点，此时这些顶点应当已经彼此孤立（第 4 行）。函数 $g(\cdot)$ 获取给定顶点集中的目标顶点。令 A'、B' 分别表示 A 和 B 中剩余的顶点（第 5–6 行）。相应地，令 C' 表示被移除的顶点，即 $C' := (A \bigcup B \bigcup C) \setminus (A' \bigcup B')$。函数 $N(\cdot,\cdot)$ 获取给定依赖子图中一个顶点集的邻居。将 C' 从 G^u 中移除后，A' 和 B' 形成两个独立子图（图 13(b)）。我们可以通过递归调用 SepPlan 依次处理这些子图（图 13(c)）（第 7–8 行）。令 $\delta(V') := |g(V')| - |s(V')|$，其中 $g(V')$ 和 $s(V')$ 分别表示顶点集 V' 中的目标顶点和起始顶点。在顶点子集 A' 和 B' 之间，我们优先处理 $\delta(\cdot)$ 值较大的那个（第 9–10 行）。这是因为，在解决由顶点集 V' 诱导的重排子问题后，缓冲区中的物体数量会减少 $\delta(V')$，或者可用的目标位姿数量会增加 $\delta(V')$。



(a) (b) (c)

图13：用于 URBM 的递归求解器 SepPlan。(a) 平面依赖图的一个 $O(\sqrt{|V|})$ 顶点分隔集。(b) 移除 C 中的起始顶点以及 C 中目标顶点的邻居后，剩余图由两个独立子图和 C 中孤立的目标位姿组成。(c) 该问题可通过递归调用 SepPlan 求解。

如附录中详细所示，SepPlan 保证了一个 MRB $\frac{20}{1-\sqrt{2/3}}\sqrt{n}$ 的上界。

定理 4. 对于具有 n 个均匀圆柱体的 *URBM*，多项式时间算法可计算出一个规划方案，其$O(\sqrt{n})$ RB，这意味着 MRB 有界于 $O(\sqrt{n})$。

6 算法

本节介绍了若干种算法，用于计算 TORO 任务在不同 MRB 目标下的解。困难性结果（定理 2）表明，计算 MRB 解的挑战在于筛选依赖图的指数数量的顶点线性排序。因此，我们提出的算法主要基于搜索，并通过许多有效的启发式方法加以增强。

我们首先介绍一种基于动态规划的方法，用于求解LRBM （第 6.1 节）。接着，我们在第 6.2 节提出一种基于优先队列的方法，用于求解 URBM。此外，我们在第 MRB 节提出了一种显著更快的动态规划深度优先改进方法，用于计算6.3。最后，我们还开发了一种整数规划模型，记为$\mathrm{TB}_{\mathrm{MRB}}$，用于在满足 MRB 约束的条件下，计算所需缓冲区总数的最小值，并将其与一种不受 MRB 约束、用于计算最小总缓冲区的算法进行比较；该算法记为 $\mathrm{TB}_{\mathrm{FVS}}$，见 [1]。第 $\mathrm{TB}_{\mathrm{MRB}}$ 节简要介绍了6.4。

6.1 用于 LRBM的动态规划（DP）

如第4.2节所述，LRBM 中的重排计划可表示为对象标签的线性排序。也就是说，给定物体的一个排序 π，我们从 $o_{\pi(1)}$ 开始。如果 $x^g_{\pi(1)}$ 未被占用，就直接将 $o_{\pi(1)}$ 移至该处。否则，将其移至一个缓冲位置。然后继续处理排序中的第二个物体

，依此类推。按照给定顺序处理完每个物体后，我们总会检查缓冲区中的物体是否可以移至其目标位置；如果有机会，就进行移动。现在，我们介绍一种用于计算此类线性排序的动态规划（DP）算法，以得到 MRB。

该算法的伪代码见算法 2。算法维护一棵搜索树 T，其中每个节点表示一种排列，在这种排列中，一组物体 S 已离开各自对应的起始位姿。我们记录当前位于缓冲区中的物体（$T[S]$.b），以及从起始布局 $\mathcal{A}_1$ 到当前布局（$T[S]$.MRB）所需的最小运行缓冲区。动态规划从一个空的 T开始。我们令根节点表示 $\mathcal{A}_1$ （第 1 行）。此时，缓冲区中没有物体，且 MRB 为 0（第 2–3 行）。随后，我们枚举所有具有 $|S| = 1$、2、 $\cdots$，最后是 n 的排列（第 4–5 行）。对于任意 S，缓冲区中的物体是在 S 中、其目标位姿仍被其他物体占据的物体（第 6 行），即 $\{o \in S | \exists o' \in \mathcal{O} \backslash S, (o, o') \in A\}$，其中 A 是 G^ℓ中的弧。 $T[S]$.MRB，从根节点 $T[\emptyset]$ 到 $T[S]$的最小运行缓冲区取决于最后一个被添加到 S中的物体 o_i，可通过枚举 o_i 计算得出（第 7–20 行）：

$$T[S].\mathrm{MRB} = \min_{o_i \in S} \max(T[S \backslash \{o_i\}].\mathrm{MRB}, \\ |T[S \backslash \{o_i\}].b| + \mathrm{TC}(S \backslash \{o_i\}, S)),$$

其中，转移代价TC 定义为

$$\mathrm{TC}(S \backslash \{o_i\}, S) = \begin{cases} 1, & o_i \in T[S].b, \\ 0, & otherwise, \end{cases}$$

当 x_i^g 当前已被占用时（第 10 行），转移代价来自依赖于 o_i 的物体：这些物体无法在将 o_i 移入缓冲区之前移出缓冲区（第 11 行）。具体而言， $T[S \backslash \{o_i\}]$.MRB 是前一个 MRB，而 $|T[S \backslash \{o_i\}].b|$ +TC($S \backslash \{o_i\}, S$) 是新转移中的运行缓冲区大小。如果以 o_i 作为从起点开始的 S 中最后一个物体时， $T[S]$.MRB 取得最小值，那么 $T[S \backslash \{o_i\}]$ 就是 T 中 $T[S]$ 的父节点（第 16-19 行）。将 $T[\mathcal{O}]$ 加入 T 后， $T[\mathcal{O}]$.MRB 就是该实例的 MRB（第 23 行），而T 中从 $T[\emptyset]$ 到 $T[\mathcal{O}]$ 的路径就是该实例对应的解。

对于 LRBM 实例（见图 3，表 1 展示了 $T[S]$.MRB 在不同最后对象选项下的情况，其中 $S = \{o_2, o_5, o_6\}$。如果最后对象 o_i 是 o_5，那么我们需要先将 o_5 移入缓冲区，然后再将 o_6 移出缓冲区。因此，尽管父节点和当前节点的缓冲区大小均为 2，但在某个时刻，三个物体会同时位于缓冲区中。然而，当我们选择 o_2 或 o_6 作为最后添加的物体时， $T[S]$.MRB 将变为 2。

6.2 一种用于 URBM 的基于优先队列的方法

与 LRBM 类似，URBM 的重排计划可以表示为 G^u 中目标顶点的线性排序。我们可以像在算法 2 中那样维护一棵搜索树，从而计算出产生 MRB 的排序。树中的每个节点 $T[g]$ 表示一种排列，其中一组目标顶点 g 已从 G^u 中移除。 $T[g]$ 的剩余依赖关系是 G^u 的一个诱导图，由 $V(G^u) \backslash (g \cup N(g, G^u))$ 构成，其中 $V(G^u)$ 是 G^u 的顶点集， $N(g, G^u)$ 是 G^u 中 g 的邻居。当目标顶点 g 从 G^u 中移除时，运行缓冲区大小为 $max(0, |N(g)| - |g|)$，即移走的物体数量减去可用目标位姿的数量。具体而言，当 $|N(g)| > |g|$时，一些物体位于缓冲区中；当 $|N(g)| < |g|$时，一些目标位姿可用；当 $|N(g)| = |g|$时，移走的物体恰好填满所有可用的目标位姿。给定一个诱导图 $I(g)$，将 $I(g)$ 中邻居不超过一个的目标顶点称为自由目标。我们有两个观察。第一，给定一个诱导图 $I(g)$，我们总可以优先移除自由目标，而不损失最优性。第二，移除一个目标顶点后，可能出现多个自由目标。例如，在图 3(a) 所示的实例中，表示 x_g^5 的顶点被移除后， x_g^2、 x_g^3 和 x_g^4 变为自由目标，可以按任意顺序加入线性排序中。将诱导图中不含自由目标的节点称为关键节点。总之，搜索树中的关键节点可能较为稀疏，而使用动态规划枚举所有可能的节点会带来很大的开销。

因此，我们不再像动态规划那样逐层探索搜索树，而是使用优先队列 Q 维护一棵稀疏树。每个节点仍表示一种排列，而树中的每条边表示清理下一个目标位姿 x_i^g 的动作。该边对应的子节点表示这种排列

算法 2： LRBM 的动态规划

输入： $G^{\ell}(\mathcal{O},\ A)$：带标签的依赖图
输出： MRB：最少运行缓冲区数量

1 $T.root \leftarrow \emptyset$
2 $T[\emptyset].b \leftarrow \emptyset$ % objects currently at the buffer
3 $T[\emptyset].\mathrm{MRB} \leftarrow 0$ % current minimum running buffer
4 **对** $1 \le k \le |\mathcal{O}|$ **执行**
  % enumerate cases where k objects have left the start poses
5 **对于** $S \in k\text{-combinations of } \mathcal{O}$ **执行**
6 $T[S].b \leftarrow \{o \in S \mid \exists o' \in \mathcal{O}\backslash S, (o,\ o')\ \in A\}$
  % Find the MRB from $T[\emptyset]$ to $T[S]$
7 $T[S].\mathrm{MRB} \leftarrow \infty$
8 **对于** $o_i \in S$ **执行**
9 $parent = S\backslash\{o_i\}$
10 **如果** $o_i \in T[S].b$ **则**
11 $RB \leftarrow \max(T[parent].\mathrm{MRB},\ |T[parent].b| + 1)$
12 **end**
13 **else**
14 $RB \leftarrow T[parent].\mathrm{MRB}$
15 **end**
16 **如果** $RB < T[S].\mathrm{MRB}$ **，则**
17 $T[S].\mathrm{MRB} \leftarrow RB$
18 $T[S].parent \leftarrow parent$
19 **end**
20 **end**
21 **end**
22 **结束**
23 **返回** $T[\mathcal{O}].\mathrm{MRB}$

表 1. $T[S].\mathrm{MRB}$ 针对不同最后对象的情况（$[p] = [parent]$）

Last object	$T[p].\mathrm{MRB}$	$T[p].b$	$T[S].b$	$T[S].\mathrm{MRB}$
o_2	1	$\{o_5\}$	$\{o_2, o_5\}$	2
o_5	2	$\{o_2, o_6\}$	$\{o_2, o_5\}$	3
o_6	2	$\{o_2, o_5\}$	$\{o_2, o_5\}$	2

其中，x_i^g 和自由目标会从诱导图中移除。与 A* 类似，我们总是从 Q 中取出具有最小 MRB 的关键节点。我们将这种基于优先队列的搜索方法记为 PQS。

6.3 深度优先动态规划

LRBM和 URBM都可以看作是在求解一系列判定问题，即询问是否能找到一个使用 $k = 1$、2、.....个运行缓冲区的重排计划。应用动态规划求解这类判定问题时，我们发现，与采用更标准的搜索树广度优先探索相比，深度优先探索的效果要好得多。我们将这种动态规划变体称为 DFDP，它对标准动态规划过程作了相当直接的调整。算法 DFDP 的整体结构见算法 3。它接收一个依赖图，并返回相应实例的最小运行缓冲区大小。本质上， DFDP 固定运行缓冲区大小 RB，并检查是否存在一个所需运行缓冲区不超过 RB 的规划方案。探索搜索树（见第 6.1 节）时，采用深度优先探索而非广度优先探索（第 4 行）。

深度优先探索的细节见算法 4。与动态规划类似，树中的每个节点都表示一种物体状态，指示物体是在起始位姿、目标位姿还是外部缓冲区中。如第 4.2 节所述，关键物体状态可以用物体集合表示 S ，这些物体已经

算法 3： 带深度优先探索的动态规划

输入： G^{ℓ}：带标签的依赖图
输出： RB：最少运行缓冲区数量

```
1 运行缓冲区← 0
2 当 not time exceeded 时
3 |  T.root ← ∅
4 |  T ← 深度优先搜索(T,∅, 运行缓冲区, G^ℓ)
5 |  如果 O ∈ T 那么返回 运行缓冲区;
6 |  否则 运行缓冲区+=1;
7 结束
```

从起始位姿处移走。如果 S 是 $\mathcal{O}$，则所有物体都处于目标位姿，我们便能在搜索树中找到从起始状态到目标状态的路径（第 1 行）。给定偏离起始位姿的物体集合 S，我们可以得到处于起始位姿的物体集合 S_S、处于目标位姿的物体集合 S_G，以及处于外部缓冲区的物体集合 S_B （第 2 行）。具体而言， $S_S = \mathcal{O} \backslash S$、 S_G 是 S 中在 S_S里没有依赖关系的物体，而 S_B 是S 中在 S_S里存在依赖关系的物体。接着，我们通过将一个物体移离起始位姿，进一步探索当前状态的子节点（第 3 行）。通过检查 G^{ℓ}（$\mathcal{O}$， A）中的依赖关系，我们确定该物体应移至缓冲区还是其目标位姿（第 4 行）。如果外部缓冲区已满，或子节点已被探索过，则从 S 到 $S \bigsqcup \{o\}$ 的转移会失败（第 5 行）。否则，我们可以将 $S \bigsqcup \{o\}$加入树中（第 7 行），并探索新节点（第 8 行）。该算法返回一棵树。

算法 4： 深度优先搜索

输入： T：搜索树； S：远离起始位姿的物体集合； RB：运行缓冲区大小； $G^{\ell}(\mathcal{O}, A)$：带标签的依赖图
输出： T

```
1 如果 S is O 则返回 T;
2 S_S, S_G, S_B = 获取状态(S)
3 对于 o ∈ S_S 执行
4 |  ToBuffer=碰撞检测(S_S\{o},o, G^ℓ)
5 |  如果 (ToBuffer and ||S_B|| + 1 > RB) or (S⊔{o} ∈ T) 则返回 T;
6 |  else
7 |  |  添加节点(T, S, S⊔{o})
8 |  |  T ← 深度优先搜索(T, S⊔{o}, 运行缓冲区, G^ℓ)
9 |  |  如果 O ∈ T 则返回 T;
10 |  end
11 结束
12 返回 T
```

直观而言，当搜索树中有许多重排计划所用的运行缓冲区不超过 k 个时，深度优先搜索会很快找到这样的解；而标准动态规划必须扩展完整棵搜索树后，才能返回一个可行解。在 [72] 中也采用了类似的深度优先探索启发式方法。

6.4 在 MRB 约束下最小化总缓冲区数

我们还构建了一个混合整数规划（MIP）模型，以最小化 MRB 或总缓冲区数。令二元变量$c_{i,\ j}$表示依赖图 G^{ℓ}：当且仅当（i， j）属于 G^{ℓ}的弧集时，$c_{i,\ j} = 1$。令$y_{i,\ j}$（$1 \leq i < j \leq n$）为二元顺序变量：当且仅当 o_i 在 o_j之前移出起始位姿时，$y_{i,\ j} = 1$ 。$y_{i,\ j}$ 用于表示动作的顺序。我们进一步引入两组二元变量$g_{i,\ j}$ 和$b_{i,\ j}$（$1 \leq i,\ j \leq n$），以表示每个时刻的物体位置。$g_{i,\ j} = 1$ 表示 o_j从起始位姿移出 o_i 时不依赖其他物体。换言之，此时 o_j的目标位姿可用。$b_{i,\ j} = 1$ 表示 o_j 从起始位置移开 o_i 后停留在缓冲区中

位姿。最后，二元变量 $B_i = 1$ 当且仅当 o_i 在某个时刻被移至缓冲区时取相应值。目标函数由两项组成：总缓冲区项和运行缓冲区项。总缓冲区项按α缩放，用于计算需要缓冲区位置的物体数量。 MRB 由整数变量 K表示，并按 β缩放。为在满足 MRB 约束的条件下最小化总缓冲区数，我们设置 $\alpha = 1$， $\beta = n$。目标函数可根据对重排计划的不同需求进行调整。具体而言，当 $\alpha = 0$、$\beta > 0$时，MIP模型最小化 MRB。当 $\alpha > 0$、 $\beta = 0$时，MIP模型最小化总缓冲区数，即重排计划中的总动作数。当 $\alpha/\beta > n - 1$时，MIP模型首先最小化总缓冲区数，然后最小化运行缓冲区数。当 $\beta/\alpha > n - 1$时，MIP模型首先最小化运行缓冲区数，然后最小化总缓冲区数。

在混合整数规划模型中，约束 2 规定了顺序变量的规则，以确保能够从 $y_{i、j}$编码出索引 1、.....、 n 的有效顺序。约束 3 表明，如果 o_j 在重排计划中曾被放入缓冲区，则 $B_j = 1$。约束4 表明， MRB K 的下界是同时放置在缓冲区中的物体数量的最大值。通过约束 5 和6，当且仅当 o_j 依赖于一个在 o_i 被移动时仍处于起始位姿的物体时， $g_{i、j} = 0$ 。通过约束7-9，当且仅当 o_j 在 o_i 之前被移动，且在 o_i 从起始位姿移出时目标位姿仍不可用时， $b_{i、j} = 1$ 。

$$\arg\min \alpha[\sum_{i=1}^{n} B_i] + \beta K \tag{1}$$

$$0 \le y_{i,j} + y_{j,k} - y_{i,k} \le 1 \quad \forall 1 \le i < j < k \le n \tag{2}$$

$$B_j \ge \sum_{1 \le i \le n} \frac{b_{i,j}}{n} \quad \forall 1 \le j \le n \tag{3}$$

$$K \ge \sum_{1 \le j \le n} b_{i,j} \quad \forall 1 \le i \le n \tag{4}$$

$$\sum_{1 \le k < i} \frac{c_{j,k}(1 - y_{k,i})}{n} + \sum_{i < k \le n} \frac{c_{j,k} y_{i,k}}{n} \le 1 - g_{i,j} \quad \forall 1 \le i, j \le n \tag{5}$$

$$1 - g_{i,j} \le \sum_{1 \le k < i} c_{j,k}(1 - y_{k,i}) + \sum_{i < k \le n} c_{j,k} y_{i,k} \quad \forall 1 \le i, j \le n \tag{6}$$

$$\frac{g_{i,j} + y_{i,j}}{2} \le 1 - b_{i,j} \le g_{i,j} + y_{i,j} \quad \forall 1 \le i < j \le n \tag{7}$$

$$\frac{g_{j,i} + (1 - y_{i,j})}{2} \le 1 - b_{j,i} \le g_{j,i} + (1 - y_{i,j}) \quad \forall 1 \le i < j \le n \tag{8}$$

$$b_{i,i} = 1 - g_{i,i} \tag{9}$$

7 应用 于有界工作空间的原地重排 e

LRBM 和 URBM 都允许利用外部开放空间进行物体的临时位移，并为带外部缓冲区的 TORO（TORE）计算重排计划。然而，在许多实际的重排场景中，物体必须在工作空间内部移动。在本节中，我们将所提出的算法应用于带内部缓冲区的 TORO（TORI）。TORI 旨在寻找一个可行的重排计划，同时最小化总动作数。求解 TORE 时只需处理由起始位姿和目标位姿定义的固有约束，而在 TORI 中，一些物体可能会在工作空间内部被临时移动，从而产生额外的已获取的约束。例如，为了解决图 1 中带外部缓冲区的问题，我们可以将 Pepsi 罐移动到外部缓冲区以打破环，先将 Coke 罐、再将 Fanta 罐移动到各自的目标位置，最后将 Pepsi 罐移回工作空间。在 TORI 中，我们必须为 Pepsi 罐在 $\mathcal{W}$ 中寻找一个临时位置。如果缓冲位置与 Coke 罐（或 Fanta 罐）的目标位置重叠，则 Coke 罐（或 Fanta 罐）又会依赖于 Pepsi 罐。为避免这些已获取的约束，缓冲位置需要避开 Coke 和 Fanta 的目标位置。由于内部缓冲区选择会产生已获取的约束，TORI（即本节研究的问题）比 TORE 更具挑战性。直观而言，在工作空间内部选择缓冲区（TORI）比使用工作空间外部的缓冲区（TORE）存放被移动的物体要困难得多，所受限制也更多。由于已有研究表明 TORE 在计算上不可处理 [1], ，且它等价于 TORI 的一个特殊情形——工作空间足够大，从而能够保证存在无碰撞的缓冲位置——因此 TORI 也是 NP难问题。此外，在 TORI 中，分配有效的缓冲区位置可能颇具挑战，因此有必要限制运行缓冲区大小。基于这一观察，我们采用 LRBM 和 URBM 的方法来计算 TORI解。

本节提出带惰性缓冲区的桌面重排（TRLB），这是一个基于第 TORI 节所述算法来求解 6 的有效框架。我们首先介绍一种采用惰性缓冲区分配的重排求解器（第 7.1 节），其中缓冲区分配会延迟到 DFDP 计算出物体移动的“粗略”调度方案之后。最后，基于 DFDP 的预处理程序可进一步加速 URBM （第 7.2 节）。为增强其在更大、更杂乱实例上的可扩展性， TRLB 框架（第 7.3 节）能够从缓冲区分配失败中恢复。

7.1 惰性缓冲区分配

物体停留在缓冲区时，应避免阻碍其他物体即将执行的操作动作。否则，缓冲区中的物体或正在操作的物体就必须让行，从而增加所需动作的数量。换言之，我们需要谨慎选择已获取的约束。如果事先知道其他物体的调度方案，就可以选择一个缓冲区，将不必要的阻碍降至最低。这一观察促使我们分两步求解重排问题：首先，计算一个原始计划，即忽略已获取约束的不完整调度方案；其次，以该不完整调度方案为参考，生成缓冲区，以优化已获取约束的选择。

7.1.1 原始计划

为了计算原始计划，我们假设有足够的空闲空间，因此不会产生任何已获取的约束。这样，问题就转化为一个 TORE 问题，其中每个物体在移动到目标位姿之前最多只会被移位一次。于是，物体 $o_i \in \mathcal{O}$ 可以有三种原始动作：

1. $(o_i, s \to g)$：从 x_i^s移动到 x_i^g；2. $(o_i, s \to b)$：从 x_i^s 移动到缓冲区；3. $(o_i, b \to g)$：从缓冲区移动到 x_i^g。

原始计划是原始动作的序列；计算这类规划方案类似于寻找依赖图的线性顶点排序 [66, 65] 。由于算法的下一阶段需要在工作空间内分配缓冲位置，因此限制需要同时分配的缓冲位置数量是有益的。我们使用 DFDP 来实现这一目标，从而最小化运行缓冲区的数量。

7.1.2 缓冲区分配

工作空间内的空闲空间 $\mathcal{W}$ 在拥挤空间中十分稀缺（例如，图17），必须通过在 $\mathcal{W}$内部谨慎分配缓冲区来处理已获取的约束。我们采用贪心策略，根据原始计划（算法5）寻找可行缓冲位。其基本思路是逐步向缓冲区添加约束，直到为整个原始计划找到可行缓冲位，或者在某一步终止，因为此时不存在适用于该原始计划的可行缓冲位。在算法5中， $\mathcal{O}_s$、 $\mathcal{O}_g$和 $\mathcal{O}_b$ 分别表示当前位于起始位姿、目标位姿和缓冲位的物体集合。

算法 5： 缓冲区分配

输入： π：一个原始计划； $\mathcal{A}_1 = \{x_1^s, ..., x_n^s\}$：起始布局； $\mathcal{A}_2 = \{x_1^g, ..., x_n^g\}$：目标布局
输出： B：缓冲区；TerminatingStep：缓冲区生成失败时的动作步骤，若成功则为 ∞ 。

1 $\mathcal{O}_s \leftarrow \mathcal{O}$; $\mathcal{O}_g, \mathcal{O}_b \leftarrow \emptyset$; $B \leftarrow$ 随机位姿($\mathcal{O}$)
2 约束 ←InitializeConstraints();
3 **对于** $(o_i, m) \in \pi$ **执行**
4 　**如果** m *is* $s \rightarrow b$ **则**
5 　　$\mathcal{O}_b$.add(o_i)
6 　　约束[o_i]←GetPoses($\mathcal{O}_s \bigcup \mathcal{O}_g - \{o_i\}$)
7 　**end**
8 　**否则如果** m *is* $b \rightarrow g$ **则**
9 　　**对于** $o \in \mathcal{O}_b \backslash \{o_i\}$ **执行** 约束[o].add(x_i^g ;
10 　　)
11 　**end**
12 **结束**
13 **否则**
14 　**对于** $o \in \mathcal{O}_b$ **执行** 约束[o].add(x_i^g ;
15 　)
16 **结束**
17 成功， $B' \leftarrow$ 缓冲区生成($\mathcal{O}_b$, 约束， B)
18 **如果** *Success*，**则**
19 　$B \leftarrow B'$
20 　$\mathcal{O}_s, \mathcal{O}_g, \mathcal{O}_b \leftarrow$ UpdateState($\mathcal{O}_s, \mathcal{O}_g, \mathcal{O}_b$)
21 **结束**
22 **否则返回** B, π.index(action);
23 **返回** B , ∞

首先，我们从 $\mathcal{A}_1$ 开始，此时所有物体都处于起始位姿，缓冲区则初始化在随机位姿处（第 1 行）。然后，我们初始化一个从 $\mathcal{O}$ 到物体位姿幂集的映射，用于指示为物体分配缓冲位姿时需要避开的障碍物集合（第 2 行）。 π 中的每个动作都表示一个被操纵的物体 o_i 以及所执行的动作 m （第 3 行）。如果 o_i 被移至缓冲区（第 4 行），我们就将其加入$\mathcal{O}_b$ （第 5 行）。 $\mathcal{O}_s \bigcup \mathcal{O}_g$ 中其他物体的当前位置被视为 o_i 的固定障碍物（第 6 行）。如果 o_i要离开缓冲区（第 8 行），那么 $\mathcal{O}_b$ 中的其他物体应避开 o_i 的目标位姿 x_i^g （第 9 行）。如果 o_i直接从 x_i^s 移动到 x_i^g （第 11 行；“else”对应于 m 处于 $s \rightarrow g$的情况，即直接从起始位姿移至目标位姿），那么当前 $\mathcal{O}_b$ 中物体的所有缓冲区都需要避开 x_i^g （第 12 行）。设置好已获取的约束后，我们通过采样或求解优化问题，为 O_b 中的物体生成新的缓冲区，以满足这些约束（第 14 行）。满足新约束的 B 中旧缓冲区将直接沿用。如果找到可行缓冲位（第 15 行），就会更新缓冲区和物体状态（第 16–17 行）。否则，我们返回计算出的可行缓冲位，并记录算法的终止步骤（第 19 行）。如果失败，返回的缓冲区会提供一个部分规划。

图 14通过一个示例说明了缓冲位姿分配过程。未着色圆盘和边界为实线的着色圆盘分别表示当前位置和目标位姿。边界为虚线的圆盘表示已分配的缓冲区。当我们将 o_1 移至缓冲区 B_1 （图 14（b））时，只需避免与 x_2^s 和 x_3^s发生碰撞。但当我们将 o_3 移至缓冲区时， B_1 还需要避开 o_3的缓冲区 B_3 。为满足新增的约束， B_1 将重新分配。由于新的缓冲区 B_1 和 B_3 （图 14（c））满足后续步骤中添加的约束，因此无需重新定位。请注意，最初的缓冲区

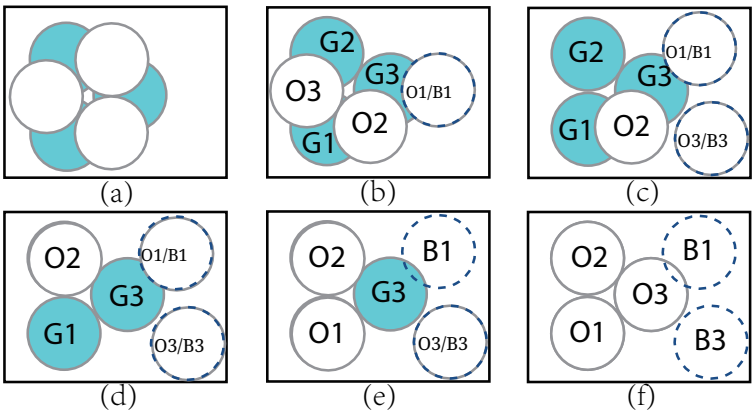


图14：在(a)中定义的三个物体的示例。原始计划为 $[(o_1, \text{s} \rightarrow \text{b}), (o_3, \text{s} \rightarrow \text{b}), (o_2, \text{s} \rightarrow \text{g}), (o_1, \text{b} \rightarrow \text{g}), (o_3, \text{b} \rightarrow \text{g})]$。图(b)-(f)展示了算法5 每执行一个动作后的步骤。边界为虚线且未着色的圆盘(B_i)表示截至每一步满足约束的缓冲区。对于每个物体 o_i，边界为实线且未着色的圆盘(O_i)表示当前位置，着色的(G_i)圆盘表示目标位姿。

选定用于 o_1 但随后被替换的物体不会出现在最终的规划方案中，即 o_1 将直接移动到新的缓冲区（图14(c)-(f)）。算法 5 每次处理依赖图中的一个强连通分量，并将其他分量中的物体视为固定障碍物。

找到可行缓冲位后，所有原始动作都可以转换为工作空间内可行的抓取放置动作。因此，原始计划可以转换为将物体从 $\mathcal{A}_1$ 移至 $\mathcal{A}_2$的重排计划。函数“缓冲区生成”通过采样或求解优化问题来实现，下面将讨论这两种方法。

采样 根据缓冲区目前需要避开的物体位姿，可以在空闲空间内通过采样位姿来生成可行缓冲位。当多个物体同时停留在缓冲区中时，我们会逐个采样缓冲区，并将先前采样得到的缓冲区视为后续缓冲区的障碍物。

优化对于半径为 r_i、位于 $(x_i,\ y_i)$ 的圆柱形物体 o_i，以及半径为 r_j、位于 $(x_j,\ y_j)$ 的圆柱形物体 o_j，当 $(x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2$ 成立时，它们之间无碰撞。进一步限制缓冲区质心的范围，以确保它们位于工作空间内，即可将缓冲区分配问题转化为二次优化问题。对于一般形状的物体，无法用物体质心的不等式来表示碰撞避免。我们可以利用物体的 ϕ 函数[73] 构建优化问题，并通过梯度求解。

7.2 预处理

在密集环境中，分配缓冲区很困难，因此需要尽量减少运行缓冲区的数量；即使在高密度设置下，URBM中的运行缓冲区数量通常也很少。基于此，对于依赖图中既不是孤立顶点也不是简单环的每个连通分量，我们首先将其作为无标签实例来求解。经过此预处理步骤后，所得有标签实例中的所有连通分量要么是简单环（$\text{MRB} = 1$），要么是孤立顶点（$\text{MRB} = 0$）。图15 展示了预处理示例。 o_1、 o_2和 o_3构成一个完全图，其中至少有两个物体需要同时放置在缓冲区中。我们通过将 o_2 移至缓冲位置、将 o_1 移至 x_3^g，并将 o_3 移至 x_2^g，对这个三顶点连通分量进行预处理。 o_2不会移至 x_1^g，因为它不占据其他目标位姿。该预处理步骤需要一个缓冲区，所得重排问题是单调的。

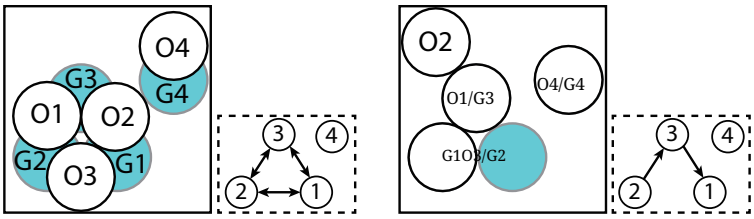


图15：一个用于预处理的四物体示例。未着色和着色的圆盘分别表示当前布局和目标布局。预处理之前（左），需要同步分配两个缓冲区。预处理之后（右），问题变为单调问题。

7.3 使用惰性缓冲区求解 TORI（TRLB）

TRLB 框架基于这样一个洞见：惰性缓冲区分配失败时，会生成一个新的 TORI 实例。新实例的目标 $\mathcal{A}_2$ 与原实例相同，但解决 TORI 任务的过程已经取得了一些进展。 TRLB有两种直接的实现方式：前向搜索和双向搜索。第一种情况下，通过接受部分解，可以构建一棵以 $\mathcal{A}_1$为根的搜索树 T ，从而计算出重排计划。在搜索树 T中，节点是可行排列，边是包含一系列无碰撞动作的部分计划。当缓冲区分配失败时，我们将由此产生的排列加入搜索树，并从 T中的一个随机节点继续重排任务。原始规划计算和缓冲区分配中的随机性，再加上这种随机性，使 TRLB 能够从失败中恢复。

算法6： TRLB 与双向搜索

输入： $\mathcal{A}_1$, $\mathcal{A}_2$, *max time*
输出： 搜索树： T_1, T_2

1 T_1.root, T_2.root← $\mathcal{A}_1$, $\mathcal{A}_2$
2 **当** *not exceeding max time* **时执行**
3 $\mathcal{A}_{rand} \leftarrow$ RandomNode(T_1)
4 $\mathcal{A}_{new1} \leftarrow$ LazyBufferAllocation($\mathcal{A}_{rand}$, T_2.root)
5 T_1.添加($\mathcal{A}_{new1}$)
6 **如果** $\mathcal{A}_{new1}$ *is* T_2.*root* **，则返回** T_1, T_2;
7 $\mathcal{A}_{near} \leftarrow$ 最近节点($\mathcal{A}_{new1}$, T_2)
8 $\mathcal{A}_{new2} \leftarrow$ LazyBufferAllocation($\mathcal{A}_{near}$, $\mathcal{A}_{new1}$)
9 T_2.添加($\mathcal{A}_{new2}$)
10 **如果** $\mathcal{A}_{new2}$ *is* $\mathcal{A}_{new1}$ **，则返回** T_1, T_2;
11 T_1, $T_2 \leftarrow T_2$, T_1
12 **结束**

在双向搜索中，以 $\mathcal{A}_1$ 和 $\mathcal{A}_2$ 为根节点的两棵搜索树会逐步构建。该过程较为复杂，如算法 6所示。该算法计算连接 $\mathcal{A}_1$ 和 $\mathcal{A}_2$的两棵搜索树。在第 1 行中，搜索树完成初始化。在每次迭代中，我们首先在随机节点 $\mathcal{A}_{rand}$ 与 T_1 的根节点之间执行重排T_2 （第 3–5 行）。LazyBufferAllocation 函数指的是第 7.1 节中提出的完整算法。找到的路径会为 TORI 生成一个可行的规划方案（第 6 行）。否则，我们会在新排列 $\mathcal{A}_{new1}$ 与其在 T_2 中的最近邻之间执行重排（第 7–9 行）。如果找到路径，则为 TORI 找到一个可行的重排计划（第 10 行）。否则，我们交换两棵搜索树，并尝试从相反方向进行重排（第 11 行）。

8 仿真性能评估

鉴于均匀圆柱体在实际应用中十分常见，我们的评估主要针对均匀圆柱体。对于仿真研究，我们创建了物体密度各不相同的实例，其密度由密度等级 $\rho := n\pi r^2/(h*w)$衡量，其中 n 是物体数量， r 是底面半径。 h 和 w 分别是工作空间的高度和宽度。换句话说， ρ 表示物体占据桌面表面的比例。

评估既针对随机物体放置，也针对人工构造的困难布局（例如，具有 $MRB = \Omega(\sqrt{n})$的依赖网格）。为了生成 ρ 值较高的测试用例，我们设计了一种基于物理引擎的方法（使用 Gazebo [74]）。我们在一个矩形盒内采样低密度的圆柱体放置位置，然后再为一些较小的“填充”物体采样位置（见图16，左）。接着，将盒子的一侧向内推，使密度达到较高水平（图16，右）；这种布局很难通过随机采样生成。通过控制两种物体的比例，可以轻松获得不同的密度等级。图17 展示了 $\rho = 0.2$，0。4 和 0。6 的三种随机物体放置布局。

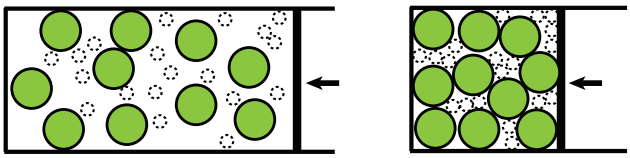


图16：通过使用基于物理引擎的模拟器将左侧场景压缩至右侧场景，生成密集实例。

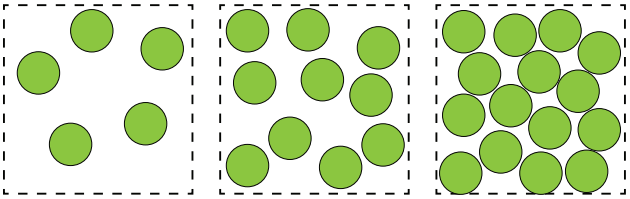


图17：无标签排列，分别对应 $\rho = 0.2$，0.4，0.6。

对于具有相同 ρ 和 n 值的两个随机生成的物体放置结果，将其中一个叠加到另一个上，即可轻松创建一个 URBM实例。通过为起始构型和目标构型中的每个物体分别赋予一个 $[n]$ 中的随机标签，可以从 URBM实例生成 LRBM实例。

对于带外部缓冲区的桌面物体重排（TORE），表2展示了不同场景（有标签或无标签）下评估的方法。对于带内部缓冲区的桌面物体重排（TORI），我们展示了在不同选项下对比惰性缓冲区生成算法的实验（表3），包括：（1）原始规划计算：采用 DFDP 的运行缓冲区最小化（RBM）、采用动态规划的总缓冲区最小化（TBM）、随机顺序（RO）；（2）缓冲区分配方法：优化（OPT）、采样（SP）；（3）高层规划器：单次运行（OS）、正向搜索树（ST）、双向搜索树（BST）；以及（4）是否进行预处理（PP）。这里，单次运行（OS）规划器使用原始计划和缓冲区分配（第7.1节），但不使用树搜索（第7.3节）。在 OS 中，我们尝试最多计算一个可行的重排计划 $30|\mathcal{O}|$ 次，之后才宣布失败。完整的 TRLB 算法由多个组件组合而成，例如，RBM-SP-BST 表示使用运行缓冲区大小最小化的原始计划、通过采样执行缓冲区分配、维护一棵双向搜索树，并且不进行预处理。

为突出 RBM 在 TORI 中的应用，我们仅比较原始规划计算中的方法，并评估预处理程序的效果。完整的消融研究见 [3]。

表 2. TORE 的评估方法。

问题	方法
LRBM	DFDP、动态规划、 $\mathrm{TB}_{\mathrm{FVS}}$、 $\mathrm{TB}_{\mathrm{MRB}}$
URBM	DFDP、 PQS

所提出的算法使用 Python 实现，所有实验均在主频为 3.00GHz 的 Intel®Xeon® 中央处理器上执行。对于求解混合整数规划，使用 Gurobi 9.16.0 [75] 。

表 3. TORI 的评估方法。

组件	方法
原始规划计算	RBM、TBM、RO
缓冲区分配方法	OPT, SP
高层规划器	OS、ST、BST
预处理	PP

8.1 LRBM在随机实例上

在图 18中，我们比较了 DP 和 DFDP在不同密度下的计算时间和成功率。每个数据点是 30 个测试用例的平均值，不包括未完成的用例（若有），每个测试用例的时间限制为 300 秒。对于 LRBM，我们能够将问题规模扩展至 $\rho = 0.4$，这是相当高的密度。结果清楚地表明， DFDP 显著优于基线DP。根据评估，两种方法均可用于解决实际规模的问题（例如，涉及数十个物体的问题），其中 DFDP展现出更高的效率和鲁棒性。

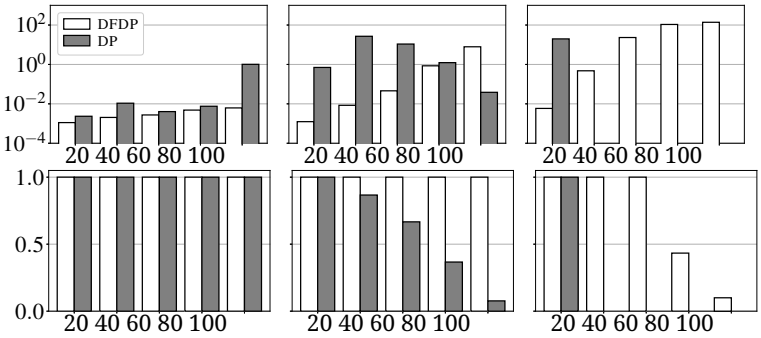


图18： DFDP 和动态规划在 LRBM 上的性能。顶行显示平均计算时间（秒），底行显示成功率；从左到右对应密度等级 $\rho = 0.2$，0。3、0。4。 x轴表示测试用例中涉及的物体数量。

同一测试用例的实际 MRB 大小见图18 左侧的图19 。我们观察到，即使对于相当大的 MRB ，其规模也很少会特别大LRBM 实例。 MRB 的规模似乎与底层依赖图的最大连通分量大小相关，见图19右侧。

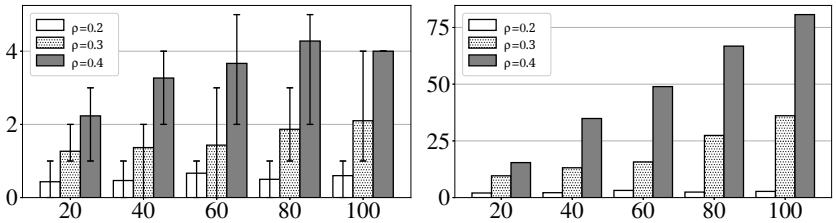


图19：对于 $\rho = 0.2$-0.4 和 $n = 20$-100 的 LRBM实例，左图显示平均值 MRB 大小及范围。右图显示依赖图中最大连通分量的大小。

对于LRBM 在 $\rho = 0.3$ 和 n 最大为 50 时，我们使用 MFVS 计算了 $\mathrm{TB}_{\mathrm{FVS}}$ 的大小（该方法无法扩展到更高的 ρ 和 n），并将其与 MRB 的大小进行了比较，如图20 (a) 所示。我们观察到， MFVS 约为 MRB的两倍，这表明 MRB 能够为估算取放系统的设计参数提供更可靠的信息。对于这些实例，我们还使用 MRB 约束计算了所需的缓冲区总数 $\mathrm{TB}_{\mathrm{MRB}}$。在约 150 个实例中，只有 1 个与 MFVS 相比有所不同（因此，图中未显示该信息）。在

图 20 (b)，我们提供了 $\mathrm{TB}_{\mathrm{FVS}}$ 和 $\mathrm{TB}_{\mathrm{MRB}}$之间的计算时间比较，结果表明， $\mathrm{TB}_{\mathrm{MRB}}$在确保达到最少运行缓冲区数量后，若希望将总缓冲区数降至最低，则具有实用性。

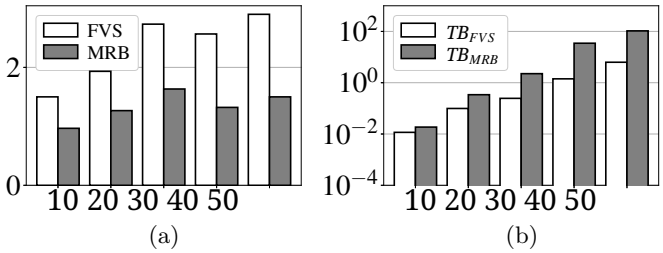


图 20: （a） MRB 和 MFVS的大小比较。（b） $\mathrm{TB}_{\mathrm{FVS}}$ 和$\mathrm{TB}_{\mathrm{MRB}}$的计算时间比较。

考虑到我们的理论发现和评估结果，可以得出一个重要结论：对于随机实例，即使实例规模很大， MRB 也实际上是一个较小的常数。此外，对于实际规模的问题，可以快速求出满足 MRB 约束时所使用的缓冲区总数的最小值。

8.2 URBM随机实例

对于 URBM，我们进行了与 LRBM 类似的性能评估。在这里，我们比较了 PQS 和 DFDP。对于 ρ 和 n的每种组合，我们都评估了 100 个随机测试用例。值得注意的是，我们可以相对轻松地达到$\rho = 0.6$ 。从图21中可以看出， DFDP 比 PQS更高效，尤其是在大规模、高密度的设置下。就 MRB 规模而言，所有受测实例的平均 MRB 规模均介于 0 和 0.7 之间，相当小（图 22）。有趣的是，随着物体数量的增加，我们观察到 MRB 有所下降，这可能是由于较大实例的“边界效应”减弱所致。也就是说，对于物体较少的实例，包围正方形对物体在其中的放置限制更多；对于较大的实例，这种限制效应会减弱。我们指出，在满足 URBM 约束的随机案例中， MRB 缓冲区总数通常非常少。

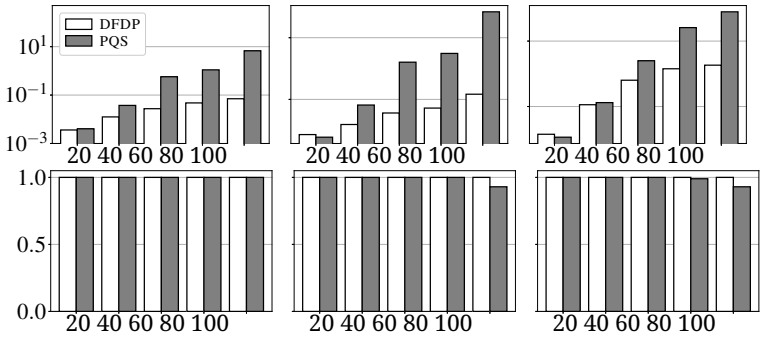


图 21: DFDP 和 PQS 在 URBM 上的性能。上排显示平均计算时间，下排显示成功率；密度等级为 $\rho = 0.4$，0。5，0。6，按从左到右的顺序排列。

8.3 手动构造的 TORE困难案例

在随机场景中，运行缓冲区大小受到限制。特别是对于LRBM，依赖图往往由多个可独立处理的强连通分量组成。我们进一步展示了 DFDP 在具有 $\mathrm{MRB} = \Theta(\sqrt{n})$ 的实例上的性能。我们评估了三类实例：(1)UG: m^2物体URBM 实例，其 G^u 依赖网格为 $\mathcal{D}(m,\ 2m)$（例如，图11）；(2) LG: m^2物体LRBM 实例，其起始与目标布局与(1)中的实例相同。(3) LC: m^2物体LRBM 实例，其中物体沿一个环放置（图12）。计算时间及相应的 MRB

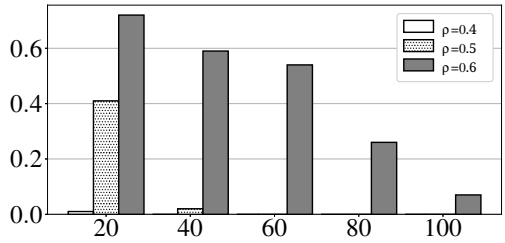


图 22：对于包含 $\rho = 0.4 - 0.6$ 和 $n = 20 - 100$的 URBM 实例， MRB 的平均值大小。当 $\rho = 0.4$ 和 0.5 时，随着物体数量超过 20， MRB 的大小接近于零。

如图23所示。对于 LG 实例，标签是随机分配的。我们进行了30个测试用例，并计算平均值后绘图。我们观察到，与密度和物体数量相近的随机实例相比，这些手工构造的实例的 MRB 大得多。

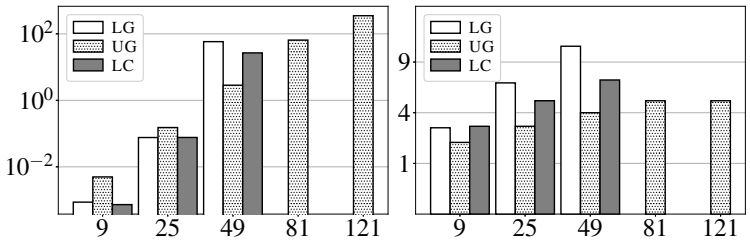


图 23：对于手工构 造的情况以及不同物体数量，左图显示了计算时间 b y DFDP 以及右 图显示了最终的 MRB 大小。

8.4 在 TORI 实例上的评估

8.4.1 圆柱形物体的消融研究。

为进行评估，我们展示了使用圆柱形物体进行的实验，以比较惰性缓冲区生成算法。消融研究的更详细版本见 [3]。我们首先比较原始规划计算选项，分别采用基于采样的缓冲区分配（SP）和双向树搜索（BST）。结果如图 24 所示。尽管 TBM-SP-BST 生成的规划方案略短于 RBM-SP-BST，但随着工作空间中的密度或物体数量增加，TBM 的可扩展性较差。与 RBM 规划相比，单个 RO 规划几乎可以瞬时生成，但我们并未发现这能显著缩短整体算法的计算时间。结果表明，在原始规划计算中尽量减少 MRB 是有益的，因为这样能够得到高效且高质量的TORI 解。

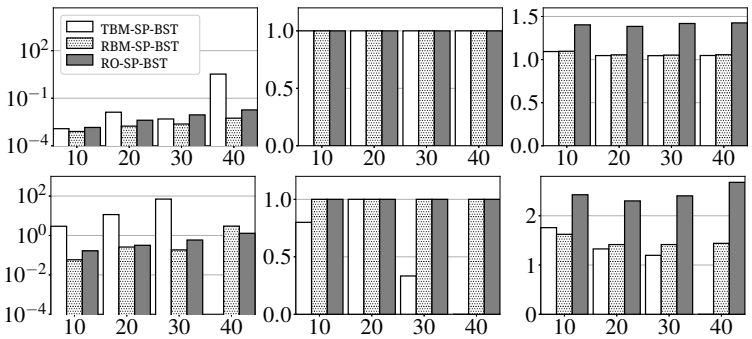


图 24：10-40 个圆柱体及密度等级 $\rho = 0.3$ （上）和 0.5（下）条件下原始规划器的比较（左：以秒为单位的计算时间；中：成功率；右：以 $|\mathcal{O}|$ 的倍数表示的动作数量）。

我们还将基于urbm的预处理程序整合到双向搜索树框架（RBM-SP-BST-PP）中，并将其与 RBM-SP-BST 进行比较。结果（图 25）表明，预处理有助于提高密集环境中的成功率。此外，在大规模密集场景中，预处理显著加快了计算速度，但代价是需要执行额外的预处理动作。通过预处理简化依赖图，可以缩短原始规划方案的计算时间。

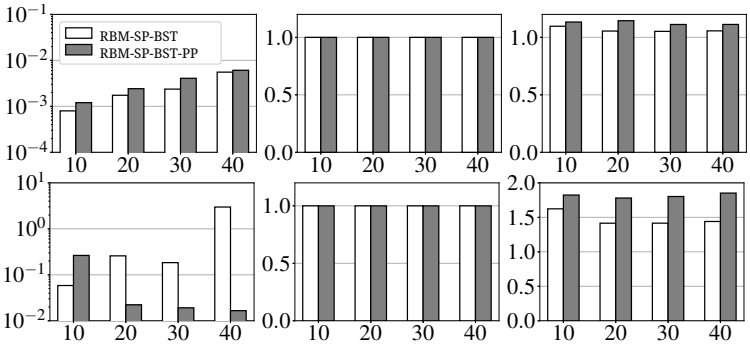


图 25：比较预处理与不进行预处理的惰性缓冲区分配算法。工作空间中有 10-40 个圆柱体，密度等级为 $\rho = 0.3$ （上）、0.5（下）（左：以秒为单位的计算时间；中：成功率；右：动作数量，为 $|\mathcal{O}|$ 的倍数）。

8.4.2 圆柱形物体与其他方法的比较。

我们将所提出的方法 RBM-SP-BST 与 BiRRT(fmRS) [49] 以及一种蒙特卡洛树搜索规划器 [76], 进行比较。据我们所知，它们是 TORI领域的最先进规划器。蒙特卡洛树搜索规划器是 C++ 求解器，而另外两种方法则使用 Python 实现。除了成功率、解的质量和计算时间之外，我们还比较了碰撞检测的次数，因为在大多数规划任务中，碰撞检测都非常耗时。在图 26中，我们比较了包含 $\rho = 0.3$ 的大规模问题中的这些方法。所有方法的成功率均为 100%。我们的 RBM-SP-BST 方法借助依赖图，避免了重复的碰撞检测。BiRRT(fmRS) 仅在局部使用依赖图，因此需要花费大量时间并进行大量碰撞检测来生成随机排列。蒙特卡洛树搜索生成的解具有相近的最优性，但同样需要进行大量碰撞检测，从而降低了计算速度。需要注意的是，右图中的数值 1（动作数量）是可能达到的最小值，因此 RBM-SP-BST 和蒙特卡洛树搜索都能计算出高质量解，而 RBM-SP-BST 的表现略好一些。总而言之，在稀疏的大规模实例中，RBM-SP-BST 的速度比其他方法快两个数量级，并且执行的碰撞检测要少得多。

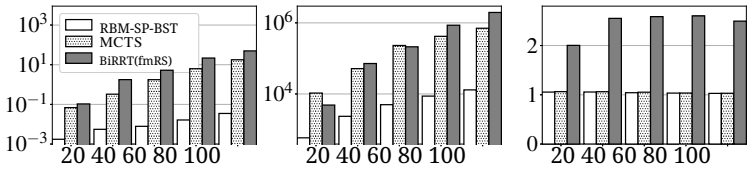


图 26：密度级别为 $\rho = 0.3$ 时，各算法在包含 20–100 个圆柱体的实例中的比较（左：计算时间，单位为秒；中：碰撞检测次数；右：动作数量，以 $|\mathcal{O}|$ 的倍数表示）。

接下来，在图 27 中，我们比较各方法在“密集小型”实例中的表现；在这些实例中，少量物体被密集地摆放在一起（图 29[左]）。在这些困难的情形下，只有 RBM-SP-BST 能保持较高的成功率。

我们进一步比较 RBM-SP-BST 和蒙特卡洛树搜索在文献 [77] 近期研究的格点重排问题中的表现。图 28[左]展示了一个包含 15 个物体的示例。在起始与目标布局中，相邻物体之间的间隙设为物体半径的 0.01 倍，因此，基于采样的方法难以分配缓冲区。蒙特卡洛树搜索会尝试每个节点上的所有动作，而 RBM-SP-BST 能够通过依赖图检测出其中隐含的组合对象关系

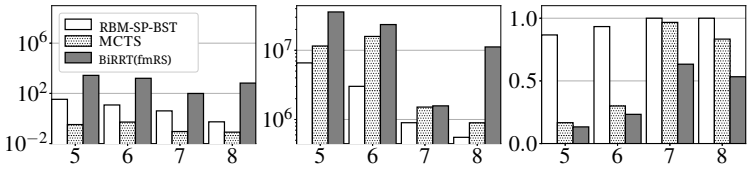


图 27：在“密集小型”实例上对各方法的比较：环境中摆放了 5–8 个物体，$\rho = 0.5$ （左：计算时间（秒）；中：碰撞次数；右：成功率）。

因此需要更少的缓冲区分配调用。如图 28[右], RBM-SP-BST 在格点重排任务中的成功率要高得多。

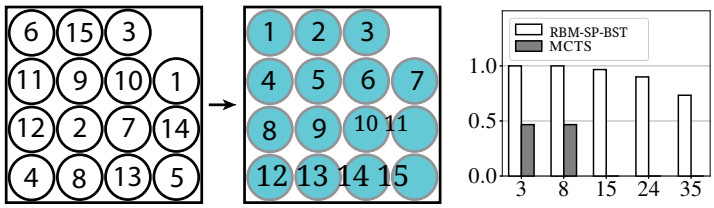


图 28：在包含 3–35 个物体的格点实例中对各方法的比较。 [左] 格点示例； [右] 成功率

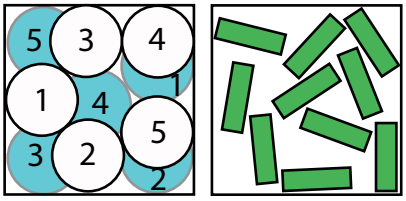


图 29： [左] “密集小型”实例：环境中摆放了 5–8 个圆柱体，密度级别为 $\rho = 0.5$。[右] 10 长方体， $\rho = 0.4$。

8.4.3 长方体对象

由于 MCTS 求解器仅支持圆柱形物体，因此在长方体设置中（图 29[右]）我们只比较 RBM-SP-BST 和 BiRRT(fmRS)。当 $\rho = 0.3$ 时，RBM-SP-BST 能够高效地计算出高质量解，而 BiRRT(fmRS) 只能求解最多包含 20 个长方体的实例。我们还指出，当 $\rho = 0.4$ 时，BiRRT(fmRS) 无法求解任何实例，而 RBM-SP-BST 平均只需 28.6 秒即可求解包含 50 个物体的重排问题。

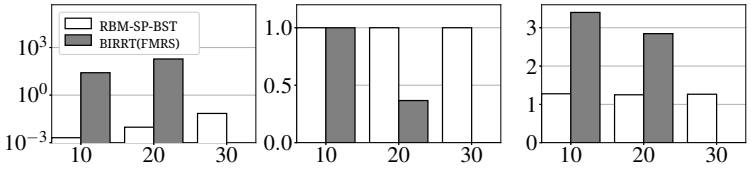


图 30：在包含 10–30 个长方体的长方体实例中对各方法的比较， $\rho = 0.3$ （左：计算时间（秒）；中：成功率；右：动作数量，以 $|\mathcal{O}|$ 的倍数表示）。

9 物理实验

本节将展示，在完整的视觉-规划-控制流程中，所提出的算法计算出的规划方案可以轻松地在真实机器人上执行。我们首先介绍硬件配置，以及执行重排计划时所采用的流程。随后，我们将介绍基于已计算出的重排计划执行情况的实验结果。

9.1 硬件配置

在我们的硬件配置中（图31），我们使用 UR-5e 机械臂搭配 OnRobot VGC 10 真空夹爪来执行抓取与放置操作。在环境上方架设了一台 Intel RealSense D435 RGB-D 相机，以俯瞰整个工作空间。我们在环境四角放置四个二维基准标记，并借助 C++ 跨平台软件库 Chilitags [78]完成相机标定。在实际机器人评估中，我们尝试了三种场景，如图 32所示。对于圆柱形物体和长方体对象，我们借助 Chilitags 进行位姿估计。对于字母物体，则使用深度学习模型 Mask R-CNN [79]估计其位姿：给定一张工作空间的俯瞰图像，预训练的 Mask R-CNN 模型会生成工作空间物体的掩码。通过计算检测到的二维点云与该物体模型点云之间的刚体变换，得到每个物体的二维位姿。

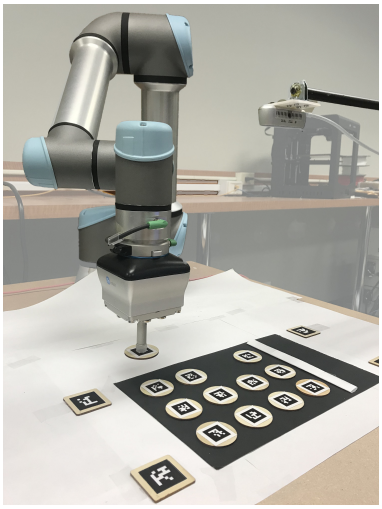


图 31：用于执行所提出的算法计算出的重排计划的硬件配置。

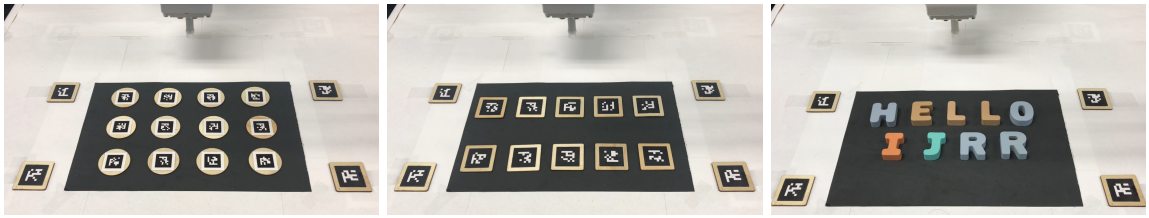


图 32：硬件平台上的三种尝试场景：圆柱体场景（左）、长方体场景（中）、字母（右）

重排流程如算法所示。7系统首先估计工作空间物体的当前位置 $\mathcal{A}_c$ （第 1 行）。给定当前布局 $\mathcal{A}_c$ 和目标布局 $\mathcal{A}_g$，重排求解器会计算重排计划 π （第 2 行）。 π 中的每个动作由三个部分组成：被操作的物体 o、当前位置 p_c和目标位姿 p_t （第 3 行）。为提高抓取与放置操作的准确性，我们会在每次抓取前更新抓取位姿（第 4–5 行）。

算法 7： 重排流程

输入： $\mathcal{A}_g$：目标布局

$\mathcal{A}_c \leftarrow PoseEstimation()$ $\pi \leftarrow$ 重排求解器$(\mathcal{A}_c, \mathcal{A}_g)$ **对于** (o, p_c, p_t) *in* π **执行** $p_c \leftarrow$ 更新位姿(o) 执行动作$((o, p_c, p_t))$ **结束**

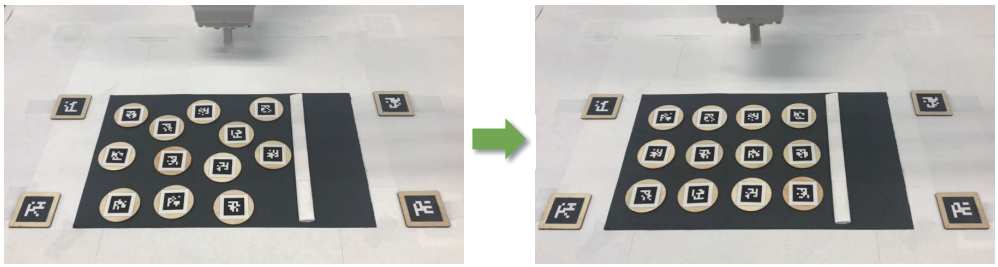


图 33：我们的实验中的一个示例实例。垫板右侧用作缓冲区放置的外部空间。

9.2 实验验证

我们在TORE上进行了硬件实验，比较 RBM 规划和 TBM 规划的执行时间。采用与第8节相同的记号，RBM 规划旨在最小化运行缓冲区大小，而 TBM 规划旨在最小化总缓冲区大小。在TORE中，最小化总缓冲区大小等同于最小化总动作数。图33展示了我们实验中的一个示例实例。垫板右侧用作缓冲区放置的外部空间。我们对 10 个包含 12 个圆柱形物体的实例进行了实验。结果见表4。在最小化运行缓冲区大小的同时，RBM 规划的平均耗时仅比 TBM 规划长 5%。

表 4。 RBM 规划和 TBM 规划在执行时间和动作数量方面的比较 s.

实例	RBM		TBM	
	执行时间（秒）	动作数	执行时间（秒）	# 动作
1	211.35	15	194.52	14
2	197.76	14	197.66	14
3	184.49	13	185.82	13
4	201.32	14	205.96	14
5	227.03	16	195.44	14
6	193.84	14	192.61	14
7	216.49	15	198.90	14
8	250.13	17	211.15	15
9	176.75	13	179.83	13
10	179.07	12	160.96	12
平均值	203.82	14.3	192.28	13.7

除了在 TORE 中进行比较外，我们还在硬件系统中比较了 TORI 算法。如配套视频所示， TRLB以显然自然且高效的方式解决了所有尝试的实例，这些实例涉及凹物体。

10 结论

在本研究中，我们探讨如何最小化运行缓冲区的数量（MRB），以通过上方抓取解决有标签和无标签桌面重排问题（TORO）；这相当于为相关底层依赖图的顶点寻找最佳线性排序。对于TORO， MRB 是一个需要理解的重要量，因为它决定了仅使用外部缓冲区时问题是否可行，而这正是某些现实应用中的情况 [1]。尽管其计算复杂性已被证明很高，我们仍提供了高效的基于动态规划的算法，能够快速计算大型且稠密的有标签/无标签TORO 实例的 MRB 。此外，我们还提供了在满足 MRB 约束的条件下最小化缓冲区总数的方法。虽然我们证明，对于均匀圆柱体的某些特殊情形，无论是有标签还是无标签设定， MRB 都可能无限增长，但实证评估表明，现实中的随机TORO 实例的 MRB 值可能小得多。我们展示了 MRB在有界受限工作空间中解决实际桌面重排问题的高度实用性。

最后，我们提出一些有趣的开放问题，供读者思考。在结构方面，尽管一般情况下已证明 LRBM 是 NP难，但均匀圆柱体情形下的 LRBM 或一般情况下的 URBM 是否在计算上难以处理，仍未有定论。至于界限，均匀圆柱体情形下 LRBM 的 MRB 下界和上界尚未吻合；能否进一步缩小界差距？物体的尺寸可能各不相同。一个有趣的问题是：将大物体还是小物体移入缓冲区更有利？移动较大的物体更具挑战性，但我们可能无需移动那么多次。

致谢

本研究部分由美国国家科学基金会项目 IIS-1845888、CCF-1934924 和 IIS-2132972 资助。我们衷心感谢匿名审稿人提出的诸多富有见地的建议和引人深思的问题，这些意见帮助提升了本研究的质量和深度。

参考文献

[1] S. D. Han、N. M. Stiffler、A. Krontiris、K. E. Bekris 和 J. Yu，“采用上方抓取进行最优桌面重排的复杂性结果与快速方法”，《国际机器人研究杂志》，第 37 卷，第 13-14 期，第 1775–1795 页，2018年。[2] K. Gao、S. W. Feng 和 J. Yu，“关于最小化桌面重排所需运行缓冲区数量”，发表于机器人学：科学与系统，2021年。[3] K. Gao、D. Lau、B. Huang、K. E. Bekris 和 J. Yu，“有界工作空间中的快速高质量桌面重排”，发表于*2022*年国际机器人与自动化会议（*ICRA*），第 1961–1967 页，2022年。[4] A. Saxena、J. Driemeyer 和 A. Y. Ng，“利用视觉进行新型物体的机器人抓取”，《国际机器人研究杂志》，第 27 卷，第 2 期，第 157–173 页，2008年。[5] M. Gualtieri、A. Ten Pas、K. Saenko 和 R. Platt，“密集杂乱环境中的高精度抓取姿态检测”，发表于*2016*年*IEEE/RSJ*智能机器人与系统国际会议（*IROS*），第 598–605 页，IEEE，2016年。[6] C. Mitash、K. E. Bekris 和 A. Boularias，“一种利用物理仿真和多视角姿态估计进行物体检测的自监督学习系统”，发表于*2017*年*IEEE/RSJ*智能机器人与系统国际会议（*IROS*），第 545–551 页，IEEE，2017年。[7] Y. Xiang、T. Schmidt、V. Narayanan 和 D. Fox，“Posecnn：一种用于杂乱场景中六维物体姿态估计的卷积神经网络”，发表于机器人学：科学与系统，2018年。

[8] M. Stilman 和 J. J. Kuffner，“可移动障碍物间导航：实时 -推理复杂环境”，《国际仿人机器人杂志》，第2卷，第04期，第479–503页，2005年。[9] K. Treleaven、M. Pavone 和 E. Frazzoli，“具有运输系统应用的单对单取送问题的渐近最优算法”，*IEEE* 自动控制汇刊，第58卷，第9期，第2261–2276页，2013年。[10] G. Havur、G. Ozbilgin、E. Erdem 和 V. Patoglu，“杂乱表面上多个可移动物体的几何重排：一种混合推理方法”，载于*2014*年 *IEEE*机器人与自动化国际会议 *(ICRA)*，第445–452页，IEEE，2014年。[11] A. Krontiris 和 K. E. Bekris，“应对困难的物体重排实例”，载于机器人学：科学与系统，第1123卷，2015年。[12] J. E. King、M. Cognetti 和 S. S. Srinivasa，“利用以物体为中心和以机器人为中心的动作空间进行重排规划”，载于*2016*年 *IEEE*机器人与自动化国际会议 *(ICRA)*，第3940–3947页，IEEE，2016年。[13] J. Lee、Y. Cho、C. Nam、J. Park 和 C. Kim，“杂乱环境中物体操作任务的高效障碍物重排”，载于*2019*年国际机器人与自动化会议 *(ICRA)*，第183–189页，IEEE，2019年。[14] R. H. Taylor、M. T. Mason 和 K. Y. Goldberg，“基于传感器的操作规划：一场与自然的博弈”，载于第四届机器人研究国际研讨会，第421–429页，1987年。[15] K. Y. Goldberg，“无传感器条件下多边形零件的定向”，*Algorithmica*，第10卷，第2期，第201–225页，1993年。[16] K. M. Lynch 和 M. T. Mason，“动态非抓持操作：可控性、规划与实验”，《国际机器人研究杂志》，第18卷，第1期，第64–92页，1999年。[17] M. Dogar 和 S. Srinivasa，“杂乱环境中推抓操作的框架”，机器人学：科学与系统 *VII*，第1卷，2011年。[18] J. Bohg、A. Morales、T. Asfour 和 D. Kragic，“数据驱动的抓取合成：综述”，*IEEE*机器人汇刊，第30卷，第2期，第289–309页，2013年。[19] N. C. Dafle、A. Rodriguez、R. Paolini、B. Tang、S. S. Srinivasa、M. Erdmann、M. T. Mason、I. Lundberg、H. Staab 和 T. Fuhlbrigge，“外在灵巧性：利用外力进行手内操作”，载于*2014*年 *IEEE*机器人与自动化国际会议 *(ICRA)*，第1578–1585页，IEEE，2014年。[20] A. Boularias、J. A. Bagnell 和 A. Stentz，“通过强化学习在杂乱环境中操纵未知物体”，载于第*29*届 *AAAI* 人工智能会议、*AAAI 2015* 及第*27*届人工智能创新应用会议、*IAAI 2015*，第1336–1342页，AI Access Foundation，2015年。[21] N. Chavan-Dafle 和 A. Rodriguez，“抓持式推送：利用推送基元进行手内操作”，载于*2015*年 *IEEE/RSJ*智能机器人与系统国际会议 *(IROS)*，第6215–6222页，IEEE，2015年。[22] L. P. Kaelbling 和 T. Lozano-P´erez，“当下的分层任务与运动规划”，载于*2011*年 *IEEE*机器人与自动化国际会议，第1470–1477页，IEEE，2011年。[23] S. Levine、C. Finn、T. Darrell 和 P. Abbeel，“ 深度视觉运动策略的端到端训练 ”，《机器学习研究期刊》，第 17 卷，第 1 期，第 1334–1373页，2016年。[24] J. Mahler、J. Liang、S. Niyaz、M. Laskey、R. Doan、X. Liu、J. A. Ojea 和 K. Goldberg，“Dex-Net 2.0：利用合成点云和解析抓取度量进行稳健抓取规划的深度学习”，*arXiv* 预印本 *arXiv:1703.09312*，2017年。

[25] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo,等, “ 通过多可供性抓取和跨域图像匹配，在杂乱环境中拾取与放置新型物体 ,” 发表于 *2018* 年 *IEEE* 机器人与自动化国际会议（*ICRA*），第3750–3757页，IEEE，2018年。[26] A. M. Wells, N. T. Dantam, A. Shrivastava 和 L. E. Kavraki，“桌面环境中任务与运动规划的可行性学习”，*IEEE*机器人与自动化快报，第4卷，第2期，第1255–1262页，2019年。[27] J. E. Hopcroft, J. T. Schwartz 和 M. Sharir，“多个独立物体运动规划的复杂性；‘仓库管理员问题’的PSPACE困难性”，《国际机器人研究杂志》，第3卷，第4期，第76–88页，1984年。[28] G. Wilfong，“存在可移动障碍物时的运动规划”，《数学与人工智能年刊》，第3卷，第1期，第131–150页，1991年。[29] M. Stilman, J.-U. Schamburek, J. Kuffner 和 T. Asfour，“可移动障碍物间的操作规划”，发表于*2007*年 *IEEE*机器人与自动化国际会议论文集，第3327–3332页，IEEE，2007年。[30] R. Wang, K. Gao, J. Yu 和 K. Bekris，“狭小空间中的惰性重排规划”，发表于国际自动规划与调度会议论文集，第32卷，第385–393页，2022年。[31] C. H. Papadimitriou，“欧几里得旅行商问题是NP完全的”，《理论计算机科学》，第4卷，第3期，第237–244页，1977年。[32] R. M. Karp，“组合问题之间的可归约性”，发表于《计算机计算的复杂性》，第85–103页，Springer，1972年。[33] S. Bereg 和 A. Dumitrescu，“重构的提升模型”，《离散与计算几何》，第35卷，第4期，第653–669页，2006年。[34] C. Nam, J. Lee, Y. Cho, J. Lee, D. H. Kim 和 C. Kim，“在杂乱且受遮挡的环境中使用机器人机械臂进行目标取回规划”，*arXiv*预印本 *arXiv:1907.03956*，2019年。[35] D. Halperin, M. van Kreveld, G. Miglioli-Levy 和 M. Sharir，“空间感知重构”，*arXiv*预印本 *arXiv:2006.04402*，2020年。[36] O. Ben-Shahar 和 E. Rivlin，“重排任务的实用推动规划”，《*IEEE*机器人与自动化汇刊》，第14卷，第4期，第549–565页，1998年。[37] E. Huang, Z. Jia 和 M. T. Mason，“大规模多物体重排”，发表于*2019*年国际机器人与自动化会议（*ICRA*），第211–218页，IEEE，2019年。[38] L. Chang, J. R. Smith 和 D. Fox，“从堆叠物体中交互式分拣物体”，发表于*2012*年 *IEEE*机器人与自动化国际会议，第3875–3882页，IEEE，2012年。[39] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan 和 K. Goldberg，“杂乱环境中的机器人抓取：利用监督者层级进行示范学习”，发表于*2016*年 *IEEE*自动化科学与工程会议（*CASE*），第827–834页，IEEE，2016年。[40] A. Eitel, N. Hauff 和 W. Burgard，“使用推动建议网络学习分离物体”，发表于《机器人研究》，第405–419页，Springer，2020年。[41]H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic 和 J. A. Stork，“使用蒙特卡洛树搜索进行多物体重排：平面非抓取式分拣案例研究”，*arXiv:1912.07024*，2019年。

[42] Z. Pan 和 K. Hauser，“用于大规模物体分拣的联合推-抓动作空间中的决策”，*arXiv*预印本 *arXiv:2010.10064*，2020年。[43] A. Zeng、S. Song、S. Welker、J. Lee、A. Rodriguez 和 T. Funkhouser，“通过自监督深度强化学习学习推与抓之间的协同作用”，载于*2018*年 *IEEE/RSJ*智能机器人与系统国际会议 *(IROS)*，第4238–4245页，IEEE，2018年。[44] B. Huang、S. D. Han、A. Boularias 和 J. Yu，“DIPN：深度交互预测网络及其在杂乱物体移除中的应用”，载于*IEEE*机器人与自动化国际会议 *(ICRA)*，2021年。[45] B. Huang、S. D. Han、J. Yu 和 A. Boularias，“利用非抓取式重排从杂乱环境中取回物体的视觉预见树”，*arXiv*预印本 *arXiv:2105.02857*，2021年。[46] E. R. Vieira、D. Nakhimovich、K. Gao、R. Wang、J. Yu 和 K. E. Bekris，“用于有效非抓取式操作的持久同调”，载于*2022*年国际机器人与自动化会议 *(ICRA)*，第1918–1924页，IEEE，2022年。[47] S. J. Buckley，多台移动机器人的快速运动规划。IBM Thomas J. Watson研究部，1988年。[48] J. van Den Berg、J. Snoeyink、M. C. Lin 和 D. Manocha，“多机器人的集中式路径规划：优化解耦为序列规划方案”，载于机器人学：科学与系统，第2卷，第2–3页，2009年。[49] A. Krontiris 和 K. E. Bekris，“高效求解一般重排任务：适用于增量式基于采样的规划器的快速扩展基元”，载于*2016*年 *IEEE*机器人与自动化国际会议 *(ICRA)*，第3924–3931页，IEEE，2016年。[50] F. Wang 和 K. Hauser，“已知物品及非确定性到达顺序下的机器人装箱”，*IEEE*自动化科学与工程汇刊，2020年。[51] K. Gao 和 J. Yu，“面向高效双臂桌面物体重排任务规划”，载于*2022*年 *IEEE/RSJ*智能机器人与系统国际会议 *(IROS)*，第10425–10431页，IEEE，2022年。[52] K. Gao 和 J. Yu，“抓取-交换式格点重排中缓冲区的效用”，*arXiv*预印本 *arXiv:2209.05390*，2022年。[53] J. D´ıaz、J. Petit 和 M. Serna，“图布局问题综述”，*ACM*计算综述 *(CSUR)*，第34卷，第3期，第313–356页，2002年。[54] M. R. Garey 和 D. S. Johnson，《计算机与不可解性》，第174卷。旧金山 Freeman 出版社，1979年。[55] C. H. Papadimitriou，“带宽最小化问题的NP完全性”，《计算》，第16卷，第3期，第263–270页，1976年。[56] M. R. Garey、D. S. Johnson 和 L. Stockmeyer，“一些简化的NP完全问题”，载于第六届 *ACM* 计算理论年度研讨会论文集，第47–63页，1974年。[57] F. Gavril，“图上的一些NP完全问题”，技术报告，计算机科学系，以色列理工学院，2011年。[58] H. L. Bodlaender、J. R. Gilbert、H. Hafsteinsson 和 T. Kloks，“树宽、路径宽度、前沿大小和最短消元树的近似”，《算法杂志》，第18卷，第2期，第238–255页，1995年。[59] T.-h. Shin、H. Oh 和 S. Ha，“最小化吞吐量约束下同步数据流图并行执行的缓冲区需求”，载于第*16*届亚洲及南太平洋设计自动化会议 *(ASP-DAC 2011)*，第165–170页，IEEE，2011年。

[60] R. J. Lipton 和 R. E. Tarjan，“平面图的分隔器定理”，《应用数学 *SIAM* 杂志》，第 36 卷，第 2 期，第 177–189 页，1979 年。[61] J. R. Gilbert、J. P. Hutchinson 和 R. E. Tarjan，“有界亏格图的分隔器定理”，《算法杂志》，第 5 卷，第 3 期，第 391–407 页，1984 年。[62] N. Alon、P. Seymour 和 P. Thomas，“具有排除小图的图的分隔器定理及其应用”，载于第二十二届 *ACM* 计算理论年度研讨会论文集，第 293–299 页，1990 年。[63] U. Elsner，《图划分——综述》。Techn. Univ.，1997 年。[64] R. Tarjan，“深度优先搜索与线性图算法”，《计算 *SIAM* 杂志》，第 1 卷，第 2 期，第 146–160 页，1972 年。[65] Y. Shiloach，“无向树的最小线性排列算法”，《计算 *SIAM* 杂志》，第 8 卷，第 1 期，第 15–32 页，1979 年。[66] D. Adolphson 和 T. C. Hu，“最优线性排序”，《应用数学 *SIAM* 杂志》，第 25 卷，第 3 期，第 403–423 页，1973 年。[67] L. M. Kirousis 和 C. H. Papadimitriou，“搜索与铺石”，《理论计算机科学》，第 47 卷，第 205–218 页，1986 年。[68] N. G. Kinnersley，“图的顶点分离数等于其路径宽度”，《信息处理快报》，第 42 卷，第 6 期，第 345–350 页，1992 年。[69] M. R. Fellows 和 M. A. Langston，“关于搜索决策与多项式时间算法的效率”，载于第二十一届 *ACM* 计算理论年度研讨会论文集，第 501–512 页，1989 年。[70] B. Monien 和 I. H. Sudborough，“对于边带权树，最小割是 NP完全的”，《理论计算机科学》，第 58 卷，第 1–3 期，第 209–229 页，1988 年。[71] F. Istv´an，“关于平面图的直线表示”，《数学科学学报》，第 11 卷，第 229–233 页，第 2 页，1948 年。[72] R. Wang、K. Gao、D. Nakhimovich、J. Yu 和 K. E. Bekris，“统一物体重排：从完备单调基元到高效的非单调知情搜索”，*arXiv* 预印本 *arXiv:2101.12241*，2021年。[73] N. Chernov、Y. Stoyan 和 T. Romanova，“物体装箱问题的数学模型与高效算法”，《计算几何》，第 43 卷，第 5 期，第 535–553 页，2010 年。[74] N. Koenig 和 A. Howard，“Gazebo（一款开源多机器人模拟器）的设计与使用范式”，载于*2004 IEEE/RSJ*智能机器人与系统国际会议（*IROS*）（*IEEE* 编目号 *04CH37566*），第 3 卷，第 2149–2154 页，IEEE，2004 年。[75] L. Gurobi Optimization，“Gurobi 优化器参考手册”，2021年。[76] Y. Labb´e、S. Zagoruyko、I. Kalevatykh、I. Laptev、J. Carpentier、M. Aubry 和 J. Sivic，“用于高效视觉引导重排规划的蒙特卡洛树搜索”，*IEEE*机器人与自动化快报，第 5 卷，第 2 期，第 3715–3722 页，2020 年。[77] J. Yu，“基于 pic-n-swaps 的格点重排：最优性结构与高效算法”，载于机器人学：科学与系统，2021年。.[78] Q. Bonnard、S. Lemaignan、G. Zufferey、A. Mazzei、S. Cuendet、N. Li、A. Ozg¨ur 和 P. Dillenbourg，“Chilitags 2：用于增强现实与机器人技术的鲁棒基准标记。”，2013 年。

[79] K. He、G.Gkioxari、P. Doll´ar 和 R. Girshick，“Mask R-CNN”，载于*IEEE*国际计算机视觉会议论文集，第 2961–2969 页，2017年。

[80] F. Fodor，“19 个全等圆在一个圆内的最密堆积”，《献给几何学》，第 74 卷，第 2 期，第 139–145 页，1999 年。

附录

特殊 TORO 设置下无标签依赖图的性质

命题的证明 2。 依赖图的最大度数等于一个物体能够与之发生重叠的互不相交物体的最大数量。在此证明中，我们通过圆盘装填问题来评估最大重叠数量的上界。我们首先讨论两个相互重叠且互不相交的正多边形之间的距离范围。随后，我们将该问题与圆盘装填问题联系起来，并推导出上界。不失一般性，我们假设正多边形的半径为 1。当两个 d边正多边形发生重叠时，两个多边形中心之间的距离不超过其外接圆半径的两倍，即 2。当两个 d边正多边形互不相交时，两个多边形中心之间的距离不小于其内切圆半径的两倍，即 $2\cos(\frac{\pi}{d})$。由于上述上界，与一个 d 边正多边形发生重叠的互不相交的 d 边正多边形的数量，不超过其中心位于一个半径为 2 的圆内的正多边形数量（图 34(a)）。此外，由于上述距离下界，该数量不超过中心位于一个半径为 2 的圆内的互不相交的 $\cos(\frac{\pi}{d})$ 圆的数量（图 34(b)），这等于装填在一个半径为 $(2+\cos(\frac{\pi}{d}))$ 的圆内的 $\cos(\frac{\pi}{d})$ 圆的数量（图 34(c)）。由于当 $d \geq 3$ 时，半径分别为 $2+\cos(\frac{\pi}{d})$ 和 $\cos(\frac{\pi}{d})$ 的圆的半径比不超过 5，因此，大圆 [80]内最多可以装填 19 个小圆。因此，无标签依赖图的最大度数不超过 19。

□

命题的证明 3。 根据工作空间中的起始位置和目标位置构造依赖图，若任意两条边都不相交，则可证明其具有平面性。反设存在两条边相交，例如图 AD 和 BC 中的 35[左侧]。由于每条边都来自一对相交的起始物体和目标物体，不失一般性，可假设 C、 D 是起始位置，而A、 B 是目标位置。因此，我们有 $|AB|$、$|CD| \geq 2r$，其中 r 是圆柱体的底面半径。由于 AD 和 BC 在依赖图中均相连，因此有 $|AD|$、$|BC| < 2r$，由此可得 $|AD| + |BC| < 4r \leq |AB| + |CD|$，这与三角不等式矛盾。因此，通过反证法证明了其平面性。

由于均匀圆柱体具有均匀的圆盘底面，一个圆柱体最多只能与六个彼此不重叠的圆柱体相切，并与至多五个彼此不重叠的圆柱体发生非平凡相交。因此，无标签依赖图的最大度数上界为 5。

□

注释。 如图所示 35[右], 命题中平面性的证明 3 不适用于具有正多边形底面的物体。

$\Omega(\sqrt{n})$ 下界 URBM

令 $(x,\ y)$ 为 $v_{x,\ y}$ 上一个顶点的坐标，其中 $\mathcal{D}(w,\ h)$ 的左上角为 $(1,\ 1)$。 $x+y$ 的奇偶性决定了该顶点所属的部集（回想一下，根据命题3，均匀圆柱体的无标签依赖图始终是平面二分图），该部集可能对应一个起始位姿或目标位姿。考虑到这一点，我们将 $\mathcal{D}(w,\ h)$ 的顶点称为起始顶点和目标顶点；不失一般性，令 $v_{1,\ 1}$ 为一个起始顶点。

我们使用 $\mathcal{D}(m,\ 2m)$ 来建立 MRB 的下界。我们使用 顶点对 $p_{i,\ j}$ 来指代两个相邻顶点 $v_{i,\ 2j-1}$ 和 $v_{i,\ 2j}$，它们位于 $\mathcal{D}(m,\ 2m)$ 中。显然，一个顶点对包含一个起始顶点和一个目标顶点。如果有物体放置在对应的目标位姿上，我们就称该目标顶点已被填充 。我们称

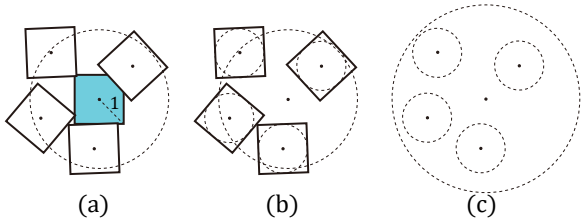


图34：(a) 与一个 d 边正多边形重叠的互不相交的 d 边正多边形数量，上界为中心位于一个 2-圆内的此类正多边形数量。(b) 中心位于一个 2-圆内的互不相交的 d 边正多边形数量，上界为中心位于一个 2-圆内的互不相交的 cos($\frac{\pi}{d}$)-圆数量。(c) 中心位于一个 2-圆内的互不相交的 cos($\frac{\pi}{d}$)-圆数量，等于填充在一个 $(2+\cos(\frac{\pi}{d}))$-圆内的 cos($\frac{\pi}{d}$)-圆数量。

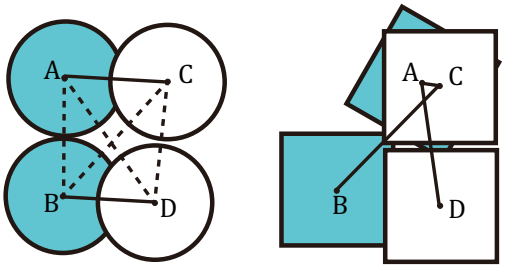


图35： [左] 命题中讨论的一个情形。3 [右] 命题 3 证明中的一个反例，当工作空间物体的底面为正方形时。

（属于某个顶点对的）起始顶点在已清除 时，表示该顶点上的对应物体已被拾取（放置到目标位置或缓冲区），但该顶点对中对应的目标尚未填充。在机器人未持有物体的任何时刻，缓冲区中的物体数量都与已清除顶点的数量相同。对于每一列 i，$1 \le i \le m$，令 f_i（resp.，c_i）表示该列中已填充的目标顶点（resp.，已清除的起始顶点）数量。请注意，只有移除对应起始顶点上的物体后，目标才能被填充。

引理 3。在依赖网格 $\mathcal{D}(m, 2m)$中，对于两列相邻列 i和 $i+1, 1 \le i < m$，如果 $f_i + f_{i+1} \neq 0$或 $2m$，则 $c_i + c_{i+1} \ge 1$。换句话说，除非 $f_i = f_{i+1} = 0$ 或 $f_i = f_{i+1} = m$，否则这两列相邻列中至少有一个已清空顶点。

证明。 如果存在 j，$1 \le j \le m$，使得顶点对 $p_{i,\ j}$ 和 $p_{i+1,\ j}$ 中的目标顶点只有一个被占据（图 36(a)），那么另一个顶点对中的起始顶点必定已清空。因此，$c_i + c_{i+1} \ge 1$。

另一方面，如果对于每个 $j, 1 \le j \le m$，$p_{i,j}$ 和 $p_{i+1,j}$ 中的目标顶点要么都被占据，要么都未被占据，那么存在 $j, 1 \le j \le m-1$，使得 $p_{i,j}$ 和 $p_{i+1,j}$ 中的目标顶点都被占据，而 $p_{i,j+1}$ 和 $p_{i+1,j+1}$ 中的目标顶点都未被占据（图 36(b)），或者情况相反（图 36(c)）。于是，对于目标顶点未被占据的顶点对，例如 $p_{i,\ j+1}$ 和 $p_{i+1,j+1}$，其中一个起始顶点是 $p_{i,j}$ 和 $p_{i,j+1}$中被占据的目标顶点的邻居。因此，$p_{i,j+1}$ 和$p_{i+1,j+1}$ 中的起始顶点至少有一个是已清空顶点。由此，$c_i + c_{i+1} \ge 1$。

□

引理 2的证明。 我们证明，当 $\lfloor n/3 \rfloor$ 个目标顶点被填充时，存在 $\Omega(m)$ 个已清除顶点。假设 q列位于 $\mathcal{D}$中，且满足 $1 \le f_i \le m-1$。根据 f_i的定义，对于这些 q列，每列至少有一个已填充的目标顶点和至少一个未填充的目标顶点。

如果 $q < \frac{\lfloor n/3 \rfloor}{3m-1}$()，那么存在两列 i和 j，使得 $f_i = m$和 $f_j = 0$。这是因为 $\sum_{1 \le i \le m} f_i = \lfloor n/3 \rfloor$ 和 $0 \le f_i \le m$ 对所有 $1 \le i \le m$都成立。因此，对于每一行中的顶点对 j，

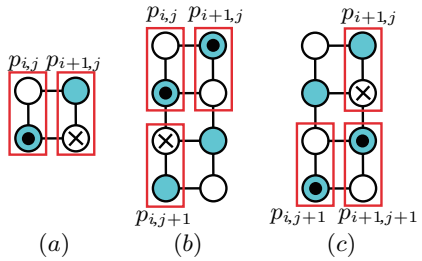


图36：引理3中讨论的一些情形。依赖图中未着色和着色的节点分别表示起始顶点和目标顶点。具体而言，内部带点的着色节点表示已填充顶点，内部带叉的未着色节点表示已清空顶点。(a) 当 $p_{i,j}$ 和 $p_{i+1,j}$ 中只有一个目标顶点被填充时，另一个顶点对中的起始顶点是已清空顶点。(b) 当 $p_{i,j}$ 和 $p_{i+1,j}$ 中的两个目标顶点都已填充，而 $p_{i,j+1}$ 和 $p_{i+1,j+1}$ 中的目标顶点都未填充时，起始顶点 $p_{i,j+1}$ 和 $p_{i+1,j+1}$ 中有一个是已清空顶点。(c) 与 (b) 相反的情形。

至少有一个目标顶点已填充，但至少有一个未填充。因此，对于每个 j，存在两列相邻列i，$i+1$，$1 \le i < m$，使得在顶点对 $p_{i,\ j}$和 $p_{i+1,\ j}$中，一个目标顶点已填充，而另一个顶点对中的起始顶点已清除（图 36(a)）。因此，这种情况下至少有 m 个已清除顶点。

如果 $q \ge \lfloor n/3 \rfloor$（），那么我们将 $\mathcal{D}$ 中的所有列划分为 $\lfloor m/2 \rfloor$ 个互不相交的列对：(1,2)、(3,4)、...

~~3 m − 1~~

...这q 列至少属于 $\lfloor q/2 \rfloor$ 对相邻列。因此，根据引理3，我们得到Θ（m）个已清除顶点。

总之，当有 $\lfloor n/3 \rfloor$ 个已填充目标时，存在 Ω(m) 个已清除顶点。因此，该实例的最小MRB 为 Ω(m)。□

$O(\sqrt{n})$ URBM的算法上界

为了研究由 SepPlan 计算出的规划方案的运行缓冲区大小，我们构造一棵基于 SepPlan 的二叉树 T（图 13(c)）。每个节点表示一次递归调用，该调用处理子图 G^u（V， E）；该节点的左、右子节点分别由 G^u（V，E）的子图 G^u（A'， E（A'））和 G^u（B'， E（B'））诱导而成。 SepPlan 通过深度优先方式遍历这棵二叉树来计算规划方案。对于 T 上的每个节点，给定输入依赖图 G^u（V， E），将处理 G^u（V， E）之前的序列记为 π_0。将 RemovalTrivialGoals(G^u（V， E）) 中被剪枝的顶点记为 P。不失一般性，假设 SepPlan 在递归调用中先处理 A'，再处理 B'。从 G^u（V， E）中分别移除 P、 C'、 A' 和 B' 中的顶点后，将所得的目标移除序列记为 π_P、 $\pi_{C'}$、 $\pi_{A'}$和 $\pi_{B'}$。我们定义函数 RB（π），用于表示移除序列 π 的“广义”当前运行缓冲区大小：当从依赖图中移除 π 中的顶点时， RB（π）等于运行缓冲区的数量，或空目标位姿数量的相反数。在深度优先递归过程中，二叉树 T 上的每个节点最多被访问三次:

1. 在探索子节点之前。峰值可能在移除 C'期间达到。令 π_{C^*} 表示运行缓冲区大小达到峰值时的序列。 $RB(\pi_{C^*}) \le RB(\pi_0) + 10\sqrt{2}\sqrt{|V|}$。2. 在处理完 A' 且尚未处理 B'之后。 $RB(\pi_{A'}) = RB(\pi_0) - \delta(C') - \delta(A')$。3. 在处理完 B'之后。 $RB(\pi_{B'}) = RB(\pi_0) - \delta(V)$。

引理4。 $RB(\pi_{A'}) \le (RB(\pi_{C^*}) + RB(\pi_{B'}))/2$。

证明。

$$
\begin{aligned}
&RB(\pi_{A'}) \\
&= RB(\pi_0) - \delta P - \delta C' - \delta A' \\
&= RB(\pi_0) - \delta(P) - \delta(C') - \max[\delta(A'), \delta(B')] \\
&\leq RB(\pi_0) - \delta(P) - \delta(C') + \frac{1}{2}[-\delta(V) + \delta(P) + \delta(C')] \\
&= \frac{1}{2}\{[RB(\pi_0) - \delta(V)] + [RB(\pi_0) - \delta(P) - \delta(C')]\} \\
&= \frac{1}{2}[RB(\pi_{B'}) + RB(\pi_{C'})] \\
&\leq \frac{1}{2}[RB(\pi_{B'}) + RB(\pi_{C^*})]
\end{aligned}
$$

□

引理4 确立了节点在 π_{C^*}、 $\pi_{A'}$和 $\pi_{B'}$ 之间的关系，位于 T上。由此引理可得每个节点的上界：

引理5. 给定二叉树 T中深度为 d的节点 N，令 $\pi_{C^*}(N)$、 $\pi_{A'}(N)$和 $\pi_{B'}(N)$分别为 N的 π_{C^*}、$\pi_{A'}$和 $\pi_{B'}$。 $RB(\pi_{C^*}(N))$、 $RB(\pi_{A'}(N))$和 $RB(\pi_{B'}(N))$均有上界

$$
\frac{[1 - (\sqrt{\frac{2}{3}})^{d+1}]}{1 - \sqrt{\frac{2}{3}}} 20\sqrt{n} \tag{10}
$$

其中 n 为该实例中的物体数量。

证明。 该结论可通过归纳法证明。

当 $d = 0$时，根节点 r 处的依赖图包含 n 个起始顶点和 n 个目标顶点。我们有$RB(\pi_{C^*}) \leq 20\sqrt{n}$， $RB(\pi_{B'}) = 0$。根据引理 4， $RB(\overline{\pi}_{A'}) \leq 10\sqrt{n}$。结论成立。

假设对于深度小于或等于 k 的所有节点，结论均成立。任取深度为 k 的节点 N，设 N 的左、右子节点分别为 L 和 R，它们的深度为 $k+1$。相应的依赖图各自至多有 $(2/3)^{k+1} \cdot 2n$ 个顶点。

$$
RB(\pi_{C^*}(L)) \leq RB(\pi_{C^*}(N)) + \sqrt{(2/3)^{k+1} \cdot 2n} \cdot 10\sqrt{2}
$$

$$
RB(\pi_{C^*}(R)) \leq RB(\pi_{A'}(N)) + \sqrt{(2/3)^{k+1} \cdot 2n} \cdot 10\sqrt{2}
$$

由于 $\pi_{B'}(L) = \pi_{A'}(N)$ 且 $\pi_{B'}(R) = \pi_{B'}(N)$，深度为 k 的节点的上界对 $\pi_{B'}(L)$ 和$\pi_{B'}(R)$均成立。因此，深度为 $k+1$ 的节点的运行缓冲区大小有一个上界

$$
\begin{aligned}
&\frac{[1 - (\sqrt{2/3})^{k+1}]}{1 - \sqrt{2/3}} 20\sqrt{n} + \sqrt{(2/3)^{k+1} \cdot 2n} \cdot 10\sqrt{2} \\
=&\frac{[1 - (\sqrt{2/3})^{k+2}]}{1 - \sqrt{2/3}} 20\sqrt{n}
\end{aligned} \tag{11}
$$

因此，上 界对所有深度为 $k+1$ 的节点均成立。由归纳法可知，引理成立。 □

根据引理 5，可以直接得出 MRB 的上界为 $\frac{20}{1 - \sqrt{2/3}}\sqrt{n}$，从而得到定理 4。