arXiv:2303.03825v1 [cs.RO] 7 Mar 2023

一种基于可达树的机器人任务与运动规划算法

Kanghyun Kim[1], Daehyung Park[2],和 Min Jun Kim[1]

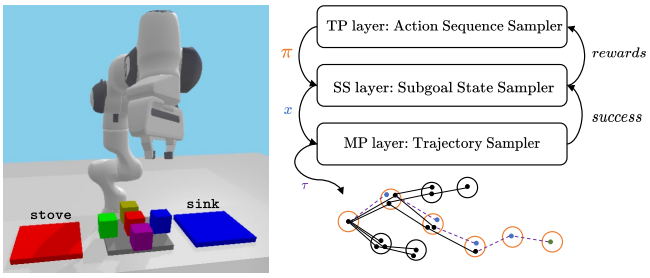
摘要—本文提出一种利用可达性树解决机器人任务与运动规划（TAMP）问题的新型算法。基于树的算法以速度快、简单易用而著称，常用于运动规划（MP），但并不适用于同时涉及抽象状态变量和几何状态变量的任务与运动规划问题。为应对这一挑战，我们提出一种分层采样策略：首先使用蒙特卡洛树搜索（MCTS）生成抽象任务计划，然后补充细节，生成几何上可行的运动轨迹。此外，我们还表明，为蒙特卡洛树搜索选择恰当的奖励，并使用预先生成且能保证几何可行的目标状态，可以显著提升所提方法的性能。通过任务与运动规划基准问题进行的对比研究，证明了所提方法的有效性。

I. 引言

目前正在开展多项研究，以使机器人能够在非结构化环境中实现自主，并执行日常任务 [1]。为此，除了低层控制算法（例如 [2], [3],）之外，机器人还应能够规划抽象的动作序列（通过任务规划/规划器，TP），同时确保每个动作在几何上的可行性（通过运动规划/规划器，MP）。值得注意的是，TP 涉及寻找离散的任务级动作序列，而 MP 涉及为机器人寻找无碰撞路径。TP 和 MP 方法均已在相关文献中得到广泛研究 [4], [5]。

为自主完成机器人操作任务，需要同时解决 TP 和 MP 问题，这被称为任务与运动规划（TAMP）。图 1 展示了一个 TAMP 问题示例，称为厨房 领域。在该问题中，机器人既可以具有几何状态变量（例如关节角度、物体位姿），也可以具有抽象状态变量（例如已烹饪 [物体]，已清洗 [物体]），而 TAMP 规划器应生成不仅满足几何约束（例如无碰撞），还满足问题中定义的任务级约束（例如所有食物块都必须在烹饪前完成清洗）的动作。这种设置的一项主要挑战是，当水槽和/或炉灶具有严格的几何约束时，规划器可能需要进行大量采样。当问题的几何结构

本研究得到韩国国家研究基金会（NRF）资助，资助来源为韩国政府（MSIT），项目编号：2021R1C1C1005232 和 2021R1A4A3032834[1]K. Kim 和 M. J. Kim 任职于电气工程学院，及 [2]D. Park 任职于韩国科学技术院（KAIST）计算机学院，韩国大田。电子邮箱：{kh11kim, daehyung, minjun.kim} at kaist.ac.kr



(a) (b)

图 1. 操作规划问题通常既涉及严格的几何约束，也涉及任务级规划。(a) 在厨房领域中，机器人需要先清洗给定的食物块，再进行烹饪。(b) 我们提出一种算法，利用三个规划层次构建具有混合状态的可达性树，以解决任务与运动规划问题。

限制了解空间时，TAMP 问题通常会变得更加复杂。在这种情况下，高层 TP 往往会提出几何上无效的动作序列。因此，底层 MP 会执行不必要的碰撞检查，从而降低整个规划器的效率。

我们的方法利用可达性树来捕捉采样状态的连通性，类似于快速探索随机树（RRT [6]）。然而，与运动规划问题不同，任务与运动规划问题具有混合状态空间，其中包含抽象状态变量和几何状态变量，这给定义两个状态之间有效的距离度量带来了挑战。因此，要将类似 RRT 的扩展方案（即随机采样一个节点并将树向其扩展）用于任务与运动规划问题并非易事。

为解决这一问题，我们提出一种分层策略：首先结合蒙特卡洛树搜索（MCTS）与符号规划器（可以是任何现成规划器）来采样抽象动作序列，随后采样相应的几何状态变量。此外，我们还提出两种提高规划速度的方法。第一，通过使用运动规划评估任务规划的结果（即动作序列），生成 MCTS 的奖励。第二，执行运动规划时，根据给定的动作序列预先生成一个目标状态候选。我们发现，预先检查目标候选的可行性可以显著提高搜索速度，尤其是在目标约束严格的问题中，例如厨房领域。为了验证我们方法的有效性，我们将其与基线方法及 PDDLStream [7]进行了比较。

II. 相关工作

与典型的运动规划问题不同，由于存在非驱动物体，操作规划的整体系统构型得以扩展。这类问题的拓扑

已通过引入模式 这一概念得到了充分确立，该概念用于参数化机器人的不同无碰撞空间。这类问题称为多模态运动规划 （MMMP） [8]。除了多模态运动规划外，TAMP 问题还需要一种更通用的设定，以考虑抽象状态和动作 [1]。也就是说，尽管多模态运动规划和 TAMP 都在处理操作规划问题，但它们的问题设定略有不同。因此，多模态运动规划和 TAMP 的求解方法往往是分别开发的。

在 TAMP 领域，通常会采用基于符号规划器的方法 [7], [9]–[13]。在大多数情况下，连续域会预先离散化或在线采样，并利用采用领域无关启发式的符号规划器搜索离散化状态空间。这些方法具有很强的通用性，因为即使面对抽象状态和动作，也能进行规划。然而，如果连续域的离散化不够密集，就可能无法找到有效的规划轨迹。反之，如果离散化过于密集，符号规划问题的规模就会变得非常大。当几何约束较为严格时，这些问题可能会变得突出。因此，随着迭代不断进行，整体规划很容易变得低效。

在 MMMP 领域，研究人员通常会采用直接处理几何信息的规划器，并保留一棵表示采样状态连通性的可达性树 [8]，[14]–[17]。该领域最著名的方法之一是类 RRT多模态运动规划方法 [18], ，它会采样一个随机状态，并将搜索树向该采样点扩展。先前研究表明，精心设计的启发式能够提升搜索速度。 [18] 利用离线计算的效用表，对有希望的转换进行偏置。在 [19], [20],中，模式转换图的边权重通过运动规划进行更新，并利用图搜索结果引导树扩展。

受前人工作的启发，本文针对任务与运动规划问题提出了一种基于可达树的方法。

III. 操作规划问题

我们通过结合运动规划、多模态运动规划和任务规划方法，为机器人操作构建了一个任务与运动规划问题。在下文中，我们首先简要定义该问题，然后详细解释符号说明。

A. 问题定义

我们将操作规划问题定义为一个元组，$(x_I,\ \mathcal{X}_G)$，其中，$x_I \in \mathcal{X}$是初始状态，$\mathcal{X}_G$是一组目标状态。该问题的目标是找到一个状态序列 $[x_0, x_1, ..., x_k]$，其中 $x_0 = x_I$、$x_k \in \mathcal{X}_G$，而 k 是状态转换的数量。本文假设系统是准静态且完全可观测的。

需要注意的是，目标集 $\mathcal{X}_G$ 可以隐式表示为某些状态变量取值构成的集合，记为 G。目标集 $\mathcal{X}_G$ 是由一组状态构成的集合，可通过对 G中的取值进行合取得到，其中未提及的状态变量可以取任意值。

B. 符号说明

1) 物体 o，可移动物体 m，机器人 r：在任务规划中，$\mathcal{O}$ 是一个物体集合，其中包含一组可移动物体 $\mathcal{M} \subset \mathcal{O}$以及机器人 $r \in \mathcal{O}$。不可移动物体也用于放置可移动物体。

2) 附着关系 α：两个物体之间的运动学关系使用附着关系定义。可移动物体的附着关系 $\alpha\ m$ 可以定义为一个元组，该元组由物体自身、父物体以及二者之间的变换组成，

$$\alpha_m = (m, p, {}^mT_p),$$
$$\text{where} \quad m \in \mathcal{M},\ p \in \mathcal{O},\ {}^mT_p \in SE(3).$$

如果运动学父节点 p 是机器人 r，则该附着关系称为抓取；否则称为放置。

3) 模式 σ：在大多数操作问题中，每个可移动物体都有一个附着关系（即一棵运动学树）。在这种情况下，某一状态下所有可移动物体的附着关系元组称为模式：

$$\sigma = \{\alpha_m | m \in \mathcal{M}\} \in \Sigma,$$

其中，模式空间 Σ 是一个包含无穷多个模式的集合。一旦模式确定，系统中所有未驱动的自由度都将被固定。也就是说，一个模式定义了机器人的无碰撞构型空间，我们将其表示为 $\mathcal{C}^{\sigma}_{free}$。

4) 抽象状态 s：规划域定义语言 PDDL [21]可用于对系统状态进行抽象，我们将其表示为 s。组成抽象状态的变量中包含如下形式的抽象附着关系，

$$\tilde{\alpha}_m = \texttt{(attached ?m ?p)},$$

它表示哪个物体附着于哪个父节点。请注意，附着关系的抽象版本仅省略两个物体之间的变换信息 mT_p 。同样，我们也可以自然地定义抽象模式 $\tilde{\sigma}$。反之，不属于抽象附着关系的变量集合定义为非几何状态 ψ。因此，$s = (\psi,\ \tilde{\sigma})$。

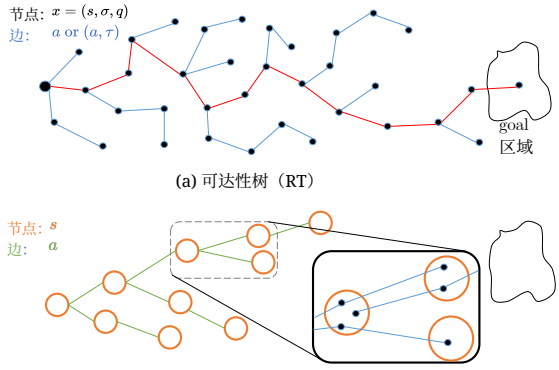
5）抽象动作 a：抽象动作空间 $\mathcal{A}$ 是一个动作集合 a，其中的动作在适用时可使抽象状态 s 发生转移。我们将适用的转移表示为：

$$s' = T(s, a).$$

与抽象状态类似，我们将能够改变抽象模式 $\tilde{\sigma}$ 的特殊类型动作$a_g \in \mathcal{A}$称为几何动作，即拾取与放置。非几何动作 $a_n \in \mathcal{A}$只能改变非几何变量，例如在厨房领域中的清洗或烹饪（见 V）。几何动作集合和非几何动作集合分别记为 $\mathcal{A}_g$ 和$\mathcal{A}_n$。

6) 混合状态 x：现在我们可以表示系统的完整状态，

$$x \in \mathcal{X} = \Psi \times \Sigma \times \mathcal{C}\,,$$



(a) 可达性树（RT）

(b) 抽象可达树（ART）

图 2. 我们的算法维护两棵树：(a) 可达性树（RT）跟踪从初始状态出发所有可达的状态；(b) 抽象可达性树（ART）是可达性树的抽象，其中每个节点包含多个 RT 节点。

其中 q 是机器人的构型，而 Ψ、Σ、$\mathcal{C}$、和 $\mathcal{X}$分别是非几何状态空间、模式空间、机器人构型空间和混合状态空间。为简便起见，我们将采用如下冗余表示，

$$x = (s, \sigma, q),$$

其中，抽象模式的信息在 s和 σ中均有冗余表示。

7) 状态转移：在两个状态 $x = (s,\ \sigma,\ q)$ 和 $x' = (s',\ \sigma',\ q')$ 之间有 3 种状态转换，每种转换都会在我们的规划器中形成三级层次结构。

- 配置转换</style>：仅 q 发生变化。也就是说，

$$||q' - q|| < \epsilon,\ \sigma = \sigma', s = s',$$

其中，$\epsilon \in \mathbb{R}$ 是一个较小的数值，表示运动连续性。

- 模式转换：仅当 q 位于由两个模式诱导的子空间中时，σ 才会发生变化，

$$\sigma \neq \sigma',\ q = q' \in \mathcal{C}^{\sigma}_{free} \cap \mathcal{C}^{\sigma'}_{free},\ \psi_s = \psi_{s'},$$

其中，$s' = T(s,\ a_g)$ $a_g \in \mathcal{A}_g$。ϕ_s和 $\phi_{s'}$分别表示 s 和 s'的非几何状态。

- 非几何状态转换：仅非几何状态 ψ 发生变化，因此，

$$\psi_s \neq \psi_{s'}\ q = q',\ \sigma = \sigma',$$

其中 $s' = T(s, a_n)$ $a_n \in \mathcal{A}_n$.

IV. 我们的规划算法*

我们的算法会不断扩展可达性树（RT），直到目标状态被纳入树中（图 2(a)）。其核心思想是通过对动作序列（即抽象规划）进行采样来扩展 RT。为此，我们使用抽象可达树 （ART），它是 RT 的一种抽象表示（图 2(b)）。由于 ART 仅考虑任务层级的信息，

*对应的视频和代码可在项目网站上获取:
https://sites.google.com/view/tree-based-tamp *

*点运算符用于在算法描述中表示节点的成员。

请注意，

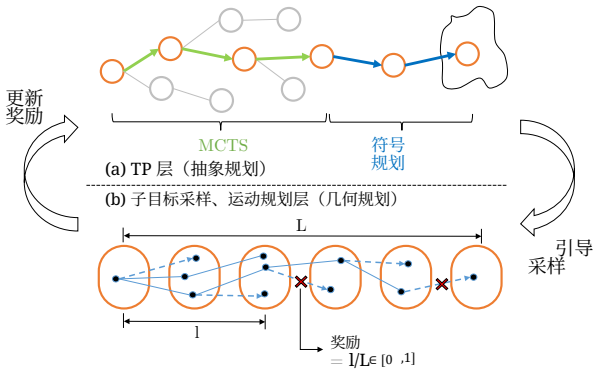


图 3. (a) TP 层利用蒙特卡洛树搜索和符号规划器对动作序列进行采样。(b) RT 扩展由采样得到的动作序列引导。当扩展失败时，SS 层会计算奖励。

一个抽象可达树节点自然包含多个可达树节点，如图 2(b) 中的黑色实线框所示。

图 3 展示了整个算法的概览。树的扩展以分层方式进行，分为三层：(1) TP 层进行动作序列采样，(2) SS 层进行子目标采样，(3) MP 层进行轨迹采样。

首先，在 TP 层中，仅使用 ART 对动作序列进行采样。动作序列采样器由随机化树搜索和符号规划器组成。也就是说，随机化树搜索获得部分规划，再由符号规划器补全剩余规划（图 3(a)）。请注意，将蒙特卡洛树搜索用作随机化树搜索，自然会在 TP 层实现启发式采样。其次，在 SS 层中，根据抽象规划（即 TP 层的输出）采样新的可达树节点以供探索。请注意，每个抽象动作都需要进行附着采样（例如，抓取动作需要抓取姿势）。SS 层以批量方式对附着关系进行采样，以预先生成一个目标状态候选项。如果该候选项不可行，则拒绝所有批量采样得到的附着关系。否则，使用采样得到的附着关系创建新的可达树节点。最后，MP 层对树节点与新节点之间的轨迹进行采样。如果 MP 失败，SS 层会计算抽象规划的成功扩展长度占比，作为奖励（图 3(b)）。随后将奖励的平均值更新至 ART，并用于在下一次迭代中引导更优抽象规划的采样。

下面对两棵主要的树进行形式化描述。

- 可达性树（*RT*） $\mathcal{G}$：每个节点对应一个混合状态 x，每条边表示两个状态之间的转换。对于几何动作，边还包含一条轨迹；对于非几何动作，边仅包含动作本身。
- 抽象可达树（*ART*） $\mathcal{T}$：每个节点对应一个抽象状态 s，每条边表示一个抽象动作 a。此外，每个节点还包含访问次数 n_{visit}、奖励总和 r_{total}以及一组可达树节点指针 V_s等信息。请注意， n_{visit} 和 r_{total} 用于 TP 层中的蒙特卡洛树搜索，而 V_s 用于 SS 层中的父节点采样。

算法 1： 任务规划层

输入： 初始状态 x_I，隐式定义的目标集 G，SS 层 迭代次数 k_{ss}
输出： 可达性树 $\mathcal{G}$

1 $\mathcal{G} \leftarrow$ 可达性树()
2 $\mathcal{T} \leftarrow$抽象状态树()
3 $\mathcal{G}$.添加节点(x_I)
4 $\mathcal{T}$.添加节点($x_I.s$)
5 **当**处于计算预算内 **时**
6 $\quad \boldsymbol{r} \leftarrow$队列()
7 $\quad \boldsymbol{\pi} \leftarrow$ 采样动作序列($\mathcal{T}, G$) // 算法2
8 $\quad \boldsymbol{n_s} \leftarrow$ 获取抽象状态节点序列($\mathcal{T}, \boldsymbol{\pi}$)
9 $\quad$ **对于** $i \leftarrow 1$ **到** k_{ss}，**执行**
10 $\quad\quad rewards \leftarrow$子目标采样层($\boldsymbol{\pi}, \boldsymbol{n_s}, G, \mathcal{G}$) // 算法3
11 $\quad\quad \boldsymbol{r} \leftarrow$拼接($\boldsymbol{r}, rewards$)
12 $\quad$ 更新树($\boldsymbol{n_s}$ 平均值($\boldsymbol{r}$))

算法 2： 动作序列采样器

输入： 抽象可达树 $\mathcal{T}$、隐式定义的目标集 G**输出：** 动作序列 $\boldsymbol{\pi}$1 n_{last}， $\boldsymbol{\pi_{rand}} \leftarrow$随机化树搜索($\mathcal{T}$)
2 $\boldsymbol{\pi_{plan}} \leftarrow$ 任务规划器(n_{last}， G)3 扩展树($\mathcal{T}$， n_{last}，$\boldsymbol{\pi_{plan}}$)4 $\boldsymbol{\pi} \leftarrow$ 拼接($\boldsymbol{\pi_{rand}}$， $\boldsymbol{\pi_{plan}}$)

A. 任务规划层（算法 *1*）

任务规划层如算法 1 和图 3(a) 所示。首先，初始化 RT $\mathcal{G}$ 和 ART $\mathcal{T}$ （第 1–4 行）。在循环中，采样动作序列（算法 2）输出一个动作序列 $\boldsymbol{\pi}$，以确保初始状态能够成功转换到抽象目标状态集。注意，抽象目标状态集可由隐式定义的目标集 G 推导得出（第 7 行）。此外，还可以使用 $\boldsymbol{\pi}$ 推导出一个抽象状态节点序列 $\boldsymbol{n_s}$（第 8 行）。子目标采样层会对这个采样序列 $\boldsymbol{\pi}$ 和 $\boldsymbol{n_s}$ 进行 k_{ss} 次评估（第 10 行）。ART $\mathcal{T}$ 根据 SS 层获得的奖励均值进行更新。

B. 动作序列采样器（算法 *2*）

动作序列采样器应提供一个随机化的抽象动作序列，引导 RT $\mathcal{G}$到达目标集 $\mathcal{X}_G$。核心思路是使用随机化树搜索创建一个部分动作序列，剩余计划则可由符号规划器完成。

对于随机化树搜索，我们采用蒙特卡洛树搜索（MCTS）机制，该机制会持续评估每个节点的价值，并在下一次迭代中输出更优的状态序列（第 1 行）。在此框架中，需要在探索（即搜索更高奖励）与利用（即利用当前信息）之间取得平衡。为此，我们在节点选择中采用 ϵ-贪心策略。也就是说，以概率 ϵ 选择子节点，否则选择价值最高的子节点。由于我们的低层规划器是半完备的，因此应当能够对相同的动作序列进行重新采样。因此，每个树节点都有概率 $\boldsymbol{\pi}$ 终止该序列。这使得随机化树搜索能够重新采样先前采样的序列。因此，

算法 3： 子目标采样层

输入： 动作序列 $\boldsymbol{\pi}$，抽象状态节点序列 $\boldsymbol{n_s}$，隐式定义的目标集 G，可达性树 $\mathcal{G}$，目标候选采样的迭代次数 采样 k_{goal}
输出： 奖励序列 $rewards$

// 目标候选生成
1 $x_I \leftarrow \mathcal{G}.root$
2 **对于** $i \leftarrow 1$ **到** k_{goal}，**执行**
3 $\quad \boldsymbol{\alpha} \leftarrow$SampleBatchAttachments($\boldsymbol{\pi}, G$) // 算法 4
4 $\quad x_G \leftarrow$ MakeGoalCandidate($x_I, \boldsymbol{\alpha}$)
5 $\quad$ **如果** 在 x_G 处没有碰撞，**则** 跳出循环
6 $\quad x_G \leftarrow$null
7 **如果** $x_G \leftarrow null$ **则返回**
// RT 的批量扩展
8 $rewards \leftarrow$ 队列()
9 **对于** $i \leftarrow 1$ **到** $\boldsymbol{\pi}.length$ **执行**
10 $\quad n \leftarrow \boldsymbol{n_s}[i]$； $a \leftarrow \boldsymbol{\pi}[i]$； $\alpha \leftarrow \boldsymbol{\alpha}[i]$； $n' \leftarrow \boldsymbol{n_s}[i+1]$
11 $\quad x \leftarrow$SampleRTNode(n) // 选择父节点
12 $\quad$ **如果** $x = null$ **则** 跳出循环
13 $\quad s \leftarrow n.s$； $\sigma \leftarrow x.\sigma$； $q \leftarrow x.q$ $s' \leftarrow n'.s$
14
15 $\quad$ **if** $a \notin \mathcal{A}_g$ **then**
$\quad\quad$ // 非几何转移
16 $\quad\quad x' \leftarrow (s', \sigma, q)$
17 $\quad$ **else**
$\quad\quad$ // 模式转换
18 $\quad\quad \sigma' \leftarrow$MakeNextMode($\sigma$， α)
19 $\quad\quad q' \leftarrow$ SampleTransition(σ， σ')
20 $\quad\quad x' \leftarrow (s', \sigma', q')$
21 $\quad success \leftarrow$MPLayer(x， x'， $\mathcal{G}$)
22 $\quad$ **如果** $success$ **那么** $n'.V_s \leftarrow n'.V_s \cup \{x'\}$
23 $\quad$ **否则** $rewards$.推送($(i-1)/\boldsymbol{\pi}.length$) // 奖励
24 **如果**上一次树扩展成功 **那么**
25 $\quad success \leftarrow$MPLayer (x'， x_G， $\mathcal{G}$)
26 $\quad$ **如果** $success$**那么**
27 $\quad\quad \mathcal{G}.solution \leftarrow x_G$ // 找到解

随机化树搜索输出随机化规划$\boldsymbol{\pi_{plan}}$ 和最终的抽象可达树节点 n_{last}。

任务规划器可以是任何现成的符号规划器（第2行）。随后，利用规划得到的序列 $\boldsymbol{\pi_{plan}}$ 扩展 ART $\mathcal{T}$（第3行）。这提高了在后续迭代中选择与当前序列 $\boldsymbol{\pi}$ 相似序列的概率。

C. 子目标采样层（算法*3*）

在 SS 层中，RT 的扩展通过三个步骤实现。首先，该层对实现所采样动作序列所需的附着关系进行批量采样。其次，利用树节点和采样得到的附着关系生成新的转移状态。最后，生成奖励信号，以向 TP 层提供信息。

SS 层首先以批量方式采样用于扩展树的所有附着关系，并基于批次样本生成目标状态候选x_G （第 1-7 行）。如果目标状态候选没有发生碰撞，则拒绝这些附着关系样本。SampleBatchAtt 的详细说明将在 IV-D 中介绍。MakeGoalCandidate 生成一个目标状态候选 $x_G = (s_G, \sigma_G, q_G)$，其中

- s_G 可以通过对初始抽象状态 $x_I.s$应用动作序列 $\boldsymbol{\pi}$ 推导得出；
- σ_G 可通过对初始模式 $x_I.\sigma$依次应用附着序列 $\boldsymbol{\alpha}$ 中的模式变化来生成；
- q_G 可以在 G中找到，也可以通过随机采样获得。

如果 x_G 无碰撞，则使用附着和目标状态进行后续树扩展过程。由于采样目标候选的计算成本很低，可以通过验证 x_G的几何可行性来拒绝无法实现的尝试。

对目标候选进行采样后，使用 $\boldsymbol{\pi}$，$\boldsymbol{n_s}$和 $\boldsymbol{\alpha}$对可达树进行批量扩展。图 3(b)展示了可达树的批量扩展过程。首先，从抽象状态节点 n 中采样一个用于扩展的父可达树节点 x （第 12 行）。如果 a 是非几何动作，则通过将抽象状态 s 更改为 s'来创建新状态 x'，这属于非几何状态转换（第 16–17 行）。如果 a 是几何动作，则通过从先前模式 σ 中更改一个附着关系 α 来生成相邻模式 σ' （第 19 行），并对 σ 和 σ' 之间的转移构型 q 进行采样（第 20 行）。请注意，这种机器人构型采样可以通过将随机构型投影到由两个模式 σ 和 σ'诱导的子空间中来实现，即数值逆运动学。

通过 MP 层检查新节点 x' 与父节点 x 之间的连接（第 22 行）。如果连接成功，则将 x' 的指针存储在下一个抽象状态节点 s' 中（第 23 行）。如果连接失败，则计算奖励信号（第 24 行）。我们将奖励定义为从根节点到当前节点的扩展长度总和，并按动作序列的长度进行缩放，使其值介于 0 和 1 之间。请注意，批量扩展过程中会收到多个奖励，因此 TP 层会根据收到的奖励的平均值进行更新。

如果动作序列 $\boldsymbol{\pi}$ 的最后一个动作成功，SS 层最终会检查最后扩展的状态 x' 与目标候选 x_G 之间的连接（第 25–26 行）。如果成功扩展到目标候选 x_G，则找到一个解（第 17–18 行）。通过回溯 RT $\mathcal{G}$，可以推导出从初始状态 x_I 到目标状态 x_G 的路径。

D. 附着关系批量采样器（算法 *4*）

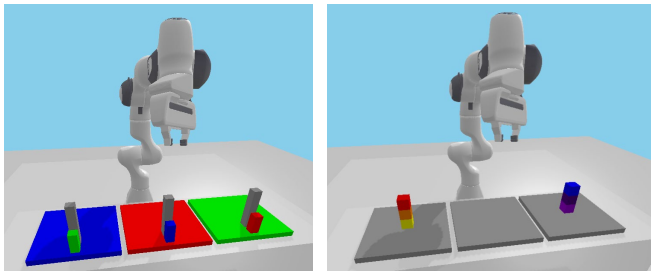
附着关系批量采样器会一次性采样动作序列 $\boldsymbol{\pi}$ 中所有必要的附着关系。值得注意的是，为了完成规划，具体动作必须从隐式定义的目标集 G 中选取相应的附着关系。我们将此动作和附着关系分别称为**目标生成动作**和**目标附着关系**。通过反向检查动作序列并识别导致最终模式的动作，可以确定目标生成动作。为此，使用集合 $\mathcal{M}_{goal}$ 来跟踪最终附着关系尚未确定的可移动物体。

该附着分配过程会忽略不需要附着的非几何动作（第 5–7 行）。如果 a 是几何动作，则需要确定其对应的附着关系。首先，我们从动作 a 中获取目标物体 m。如果物体 m 尚未分配附着关系，且动作 a 在隐式定义的目标集中具有对应的目标

算法 4： 附着关系批量采样器

输入： 动作序列 $\boldsymbol{\pi}$，隐式定义的目标集 G
输出： 附着序列 $\boldsymbol{\alpha}$

```
1  α ← 队列()
2  M_goal ← M
3  对于 i ← π.长度 到 1 执行
4      a ← π[i]
5      如果 a ∉ A_g 则
6          α.推送(空)
7          继续
8      m ←目标可移动物体(a)
9      如果 m ∈ M_goal 且 是否为目标附着(a, G)
         then
10         α_m ←获取目标附着(a, G)
11     else
12         α_m ← 采样附着物(a)
13     α.推(α_m)
14     M_goal ← M_goal − {m}
15 α ←反转(α)
```



(a) (b)

图 4. (a)非单调领域。(b) 积木塔领域。

附着关系 G，则可以得出动作 a 是目标生成动作。在这种情况下，我们不进行采样，而是直接从集合 G 中选择目标附着关系（第 9–10 行）。如果该动作与目标没有直接关联，则根据该动作对附着关系进行采样（第 11–12 行）。

V. 实证评估

本节通过三个任务与运动规划基准问题验证所提出的方法：

- **厨房** m **领域** *[22]*：我们提供 m 个可移动食物块和三个可放置物体（即盘子、水槽和炉灶），如图 1 所示。任务是烹饪每个最初放在盘子上的食物块。食物块必须先清洗才能烹饪。因此，有两个非几何动作：烹饪和清洗。当食物块分别放在炉灶和水槽上时，可以执行相应的动作。该领域具有严格的几何约束，即水槽和炉灶的区域很小，因此积木只能紧密地摆放。

- **非单调** m **领域** *[23]*：我们提供 m 不同颜色的积木和 m 高度更高的阻挡块。机器人任务是将彩色方块放置在地面上，方块的颜色必须与其目标位置的颜色相同。如图 4(a) 所示，难点在于机器人无法立即够到彩色方块。因此，灰色阻挡块

必须先移除，才能接近彩色方块。然而，由于任务规划器不了解此类几何信息，因此 TP 层与较低层之间的交互至关重要。

- 积木塔 m 领域 *[24]*：我们提供 m 块积木和3个可放置的盘子。初始时，我们将积木以随机顺序堆叠在左侧或右侧的盘子上，如图4(b)所示。机器人任务是按照特定顺序将它们堆叠在中间的盘子上。为处理可移动物体可以放置在另一个可移动物体上的任务约束，我们引入了两个抽象动作，即堆叠和拆堆。

作为比较，我们实现了PDDLStream，这是在线上最知名且最容易获取的TAMP求解器。此外，我们还设计了两个基线规划器：

- 无奖励规划器：我们的规划器利用奖励作为偏置，以改进动作序列采样。为验证这一做法的有效性，此基线不使用奖励。
- 无拒绝规划器：我们的规划器通过生成目标候选并检查其几何可行性来拒绝样本批次。为验证这一过程的有效性，此基线不检查目标状态的可行性。

所有规划器（包括 PDDLStream）均使用 Python 编写。对于所提出的方法和基线规划器，抽象领域采用 PDDL 建模，任务计划由 Pyperplan 符号规划器求解 [25]。对于 MP 层，我们使用了 RRTConnect [26]的实现。碰撞检测和逆运动学通过 Pybullet 物理库实现 [27]。所有算法均在配备 Intel Core i7-10700 CPU 和 16Gb 内存的 PC 上运行。所提出的方法（包括基线方法）使用了以下超参数：$k_{ss} = 2$、$k_{goal} = 10$和$\epsilon = 0.5$。对于 PDDLStream，我们使用了已知表现最佳的实现 *adaptive*，并采用默认参数。所有实验均重复进行 30 次，超时时间设为 100 秒。

A. 讨论

仿真结果以累积分布函数的形式呈现在图 5 中。在厨房领域，选择简单的任务计划是更好的策略。例如，按顺序完成清洗-烹饪任务更有优势，因为水槽的目标区域较小。我们的规划器成功率为 97%，而无拒绝基线的成功率为 90%，PDDLStream 和无奖励基线的成功率约为 70%。

比较两种基线方法的性能时，无利用目标候选采样的奖励基线，在 kitchen 3 和 4 域中的表现优于无拒绝基线。这直观上是合理的，因为这些域中的积木稀疏性使随机采样得到的抽象规划很可能可行。相比之下，在 kitchen 5 中，无拒绝基线（即由先前奖励提供信息的规划器）表现更好。所提出的规划器

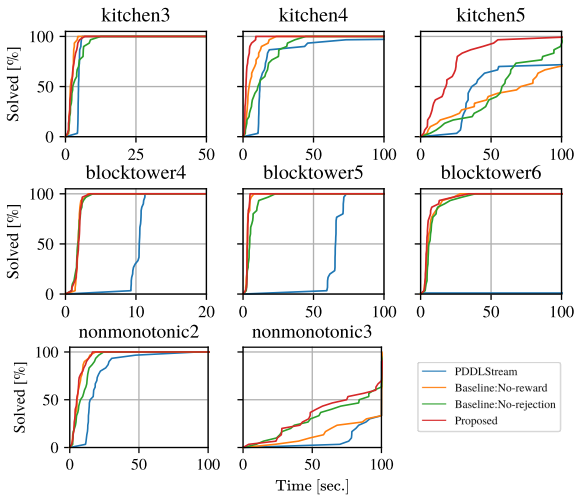


图 5. 所提出的规划器、两种基线规划器（无奖励、无拒绝）和 PDDLStream 的比较结果。结果以累积分布函数表示。

通过同时利用这两项特性，在所有场景中都取得了最高性能。

奖励的有效性在非单调领域中也能体现出来，在这些领域中，必须先移除阻挡方块。在所提出的方法中，任务规划器通过奖励学习到，移除阻挡方块的动作序列更有希望。因此，在非单调领域中，我们提出的方法表现最佳，成功率达到 66%。在该领域中，拒绝使用目标候选项并未显著提升性能。

最后，在积木塔领域中，所提出的方法（包括基线方法）均优于 PDDLStream。这是因为与 PDDLStream 相比，所提出的方法要解决的符号规划问题简单得多。事实上，[24] 已经报告称，对于 PDDLStream 而言，包含 4 个以上物体的积木塔领域具有挑战性。在积木塔6的案例中，PDDLStream 未能找到任何解，而所提出的方法在 13 秒内达到了 90% 的成功率。与 PDDLStream 相比，我们的规划器将抓取、放置、构型和轨迹等几何规划实例存储在可达性树中，而不是将其纳入任务规划问题。换言之，我们的方法使任务规划问题保持在可处理的范围内，并防止其随着时间推移因采样实例而变得更加复杂。

六、结论

本文提出了一种基于树的TAMP求解器，其灵感来自基于可达性树的MMMP求解器。所提出的方法对抽象动作序列进行采样，使树扩展不会违反抽象领域中定义的任务约束。通过几何规划器评估采样得到的序列，可以引导搜索转向更有希望的抽象状态区域。此外，还可以采用一种预先生成目标候选的采样拒绝方案，有效处理严格的目标约束。通过与两种基线方法及 PDDLStream 进行比较研究，验证了我们方法的有效性。

参考文献

[1] C. R. Garrett、R. Chitnis、R. Holladay、B. Kim、T. Silver、L. P. Kael- bling 和 T. Lozano-P´erez，《集成任务与运动规划》，《控制、机器人与自主系统年度评论》，第4卷，第265–293页，2021年。[2] M. J. Kim、A. Werner、F. Loeffl 和 C. Ott，《通过内环力矩控制实现带黏弹性关节机器人的被动阻抗控制》，《IEEE机器人学汇刊》，第38卷，第1期，第584–598页，2021年。[3] J. Jeong、H. Mishra、C. Ott 和 M. J. Kim，《一种基于记忆的SO(3)参数化：径向无界势函数下六维阻抗控制的理论与应用》，载于《IEEE国际机器人与自动化会议论文集》，2022年，第8338–8344页。[4] S. M. LaValle，《规划算法》。剑桥大学出版社，2006年。[5] M. Ghallab、D. Nau 和 P. Traverso，《自动规划：理论与实践》。Elsevier，2004年。[6] S. M. LaValle，《快速探索随机树：一种新的路径规划工具》，年度研究报告，1998年。[7] C. R. Garrett、T. Lozano-P´erez 和 L. P. Kaelbling，《PDDLStream：通过乐观自适应规划集成符号规划器与黑盒采样器》，载于《国际自动规划与调度会议论文集》，第30卷，2020年，第440–448页。[8] K. Hauser 和 J.-C. Latombe，《非扩张空间中的多模态运动规划》，《国际机器人研究期刊》，第29卷，第915–897页，2010年，第7期。[9] C. R. Garrett、T. Lozano-P´erez 和 L. P. Kaelbling，《用于因子化任务与运动规划的基于采样的方法》，《国际机器人研究期刊》，第37卷，第13–14期，第1796–1825页，2018年。[10] S. Srivastava、E. Fang、L. Riano、R. Chitnis、S. Russell 和 P. Abbeel，《通过可扩展的规划器无关接口层实现任务与运动规划》，载于《IEEE国际机器人与自动化会议论文集》，2014年，第639–646页。[11] T. Lozano-P´erez 和 L. P. Kaelbling，《一种用于解决序贯操作规划问题的基于约束的方法》，载于《IEEE/RSJ智能机器人与系统国际会议论文集》，2014年，第3684–3691页。[12] L. P. Kaelbling 和 T. Lozano-P´erez，《当下的分层任务与运动规划》，载于《IEEE国际机器人与自动化会议论文集》，2011年，第1470–1477页。[13] N. T. Dantam、Z. K. Kingston、S. Chaudhuri 和 L. E. Kavraki，《用于任务与运动规划的增量式约束框架》，《国际机器人研究期刊》，第37卷，第10期，第1134–1151页，2018年。[14] J. Mirabel 和 F. Lamiraux，《操作规划：解决交叉叶状结构问题》，载于《IEEE国际机器人与自动化会议论文集》，2017年，第4032–4037页。[15] T. Sim´eon、J.-P. Laumond、J. Cort´es 和 A. Sahbani，《基于概率路图的操作规划》，《国际机器人研究期刊》，第23卷，第7–8期，第729–746页，2004年。[16] K. Hauser、T. Bretl、J.-C. Latombe、K. Harada 和 B. Wilcox，《多样地形上的足式机器人运动规划》，《国际机器人研究期刊》，第27卷，第11–12期，第1325–1349页，2008年。[17] J. Barry、L. P. Kaelbling 和 T. Lozano-P´erez，《采用多样化动作的分层操作方法》，载于《IEEE国际机器人与自动化会议论文集》，2013年，第1799–1806页。[18] K. Hauser 和 V. Ng-Thow-Hing，《面向人形机器人操作任务的随机化多模态运动规划》，《国际机器人研究期刊》，第30卷，第6期，第678–698页，2011年。[19] Z. Kingston 和 L. E. Kavraki，《扩展多模态规划的规模：利用经验并为离散搜索提供信息》，《IEEE机器人学汇刊》，2022年。[20] Z. Kingston、A. M. Wells、M. Moll 和 L. E. Kavraki，《利用协同离散引导为多模态规划提供信息》，载于《IEEE国际机器人与自动化会议论文集》，2020年，第3199–3205页。[21] D. McDermott、M. Ghallab、A. Howe、C. Knoblock、A. Ram、M. Veloso、D. Weld 和 D. Wilkins，《PDDL——规划域定义语言》，1998年。[22] T. Ren、G. Chalvatzaki 和 J. Peters，《用于机器人任务与运动规划的扩展树搜索》，arXiv预印本 arXiv:2103.05456，2021年。[23] F. Lagriffoul、N. T. Dantam、C. Garrett、A. Akbari、S. Srivastava 和 L. E. Kavraki，《任务与运动规划的平台无关基准》，《IEEE机器人与自动化快报》，第3卷，第3765–3772页，2018年，第4期。[24] M. Khodeir、B. Agro 和 F. Shkurti，《利用流学习任务与运动规划中的搜索》，arXiv预印本 arXiv:2111 13144，2021年。[25] Y. Alkhazraji、M. Frorath、M. Griitzner、M. Helmert、T. Liebetraut、R. Mattmiiller、M. Ortlieb、J. Seipp、T. Springenberg、P. Stahl 等，《Pyperplan》，2020年，网址：https://doi.org/10.5281/zenodo.3700819。

[26] J. J. Kuffner 和 S. M. LaValle，“RRT-connect：一种高效的单查询路径规划方法”，载于 *IEEE* 国际机器人与自动化会议论文集，第 2 卷，2000 年，第 995–1001 页。[27] E. Coumans 和 Y. Bai，“PyBullet：一个用于游戏、机器人学和机器学习物理仿真的 Python 模块”，*2016*，网址 *http://pybullet.org*。