arXiv:2303.00637v1 [cs.RO] 1 Mar 2023

通过主动复用验证工作量，高效解决操作规划问题中的路径规划

Valentin N. Hartmann[1,2],Joaquim Ortiz-Haro[2],Marc Toussaint[2]

摘要——操作规划以及任务与运动规划场景中产生的路径规划问题通常具有重复性：同一机械臂在变化很小的空间中移动。尽管信息复用潜力很大，却很少有规划器能充分利用现有信息。为更好地实现复用，我们将碰撞检测分解为可复用部分和不可复用部分。随后，我们将操作规划中的路径规划问题序列视为多查询路径规划问题。这使我们能够使用在多个查询间主动减少规划工作量的规划器，从而主动复用先前的知识。

我们在 EIRM* 和按工作量排序的 LazyPRM* 中实现了这一方法，并在多个模拟机器人示例上对其进行了基准测试。此外，我们还表明，对碰撞检测进行分解的方法还能让所获得的知识在同一问题的多个不同实例间复用，即应用于多查询操作规划场景。

采用分解式碰撞检测的规划器，初始解时间最多可比其他规划器快一倍，同时提供相近的解的质量。

I. 引言

操作规划既需要规划高层决策（即采取哪些动作，例如“抓取 obj_1 —放置 obj_1”），也需要进行低层规划，包括规划抓取位置，以及起始和目标构型之间的运动规划。在本研究中，我们聚焦于操作规划问题中的运动规划部分。在求解此类操作规划问题中的路径规划问题时，方法的快速性和可靠性至关重要。在路径规划中，检查所提出的路径是否无碰撞通常是最耗时的部分 [1]，[2]，，尤其是在寻找初始解时 [3]。

操作规划问题通常按顺序求解。也就是说，在上述示例中，规划器首先为“抓取”动作计算一条路径，随后再为“放置”动作计算路径。这意味着可以重复利用先前已解决的规划查询所获得的信息。为了尽量缩短整体规划时间，我们希望主动 重复利用当前及先前规划查询过程中获得的信息。然而，无法直接重复利用有关运动有效性或无效性的信息，因为规划期间环境可能发生变化（例如移动一个

本研究得到德国研究联合会（DFG，德国研究基金会）德国卓越战略项目 EXC2120/1–390831618 的资助，并获得德以科学研究基金会（GIF）资助，项目编号为 I-1491-407.6/2019。Joaquim Ortiz-Haro 感谢马克斯·普朗克智能系统国际研究生院提供的支持。

[1]VISUS，德国斯图加特大学{firstname}.{lastname}@ipvs.uni-stuttgart.de[2]德国柏林工业大学学习与智能系统组

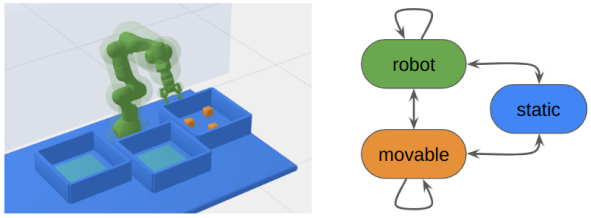


图1：**左图**：分解式碰撞检测中的碰撞对象示意图。蓝色部分均为静态对象，橙色部分均为可移动对象，绿色部分均属于机器人。**右图**：箭头连接了需要彼此进行碰撞检测的各组对象。

障碍物），因此机器人（通过抓取并搬运物体）有效的碰撞形状可能发生变化，先前有效的运动可能变得无效，反之亦然。

在大多数用于操作规划的规划器中，先前规划查询过程中获得的知识都会被丢弃 [4]（即，规划器已经规划好抓取物体的运动后，在搜索将物体放置到某处的路径时，会从头开始），或者仅用于引导新的查询中的规划过程 [5]。实际上，规划器需要反复重新发现并验证相同的边，浪费了计算开销。

我们提出了一种方法，使碰撞信息能够在整个操作规划问题中得到重复利用。随后，我们将单个操作规划问题中的一系列运动规划问题视为一个多查询运动规划问题。这样，我们便能重复利用先前规划查询中获得的碰撞信息。我们在两个能够主动重复利用工作量的规划器中实现了这一方法：努力知情路线图（EIRM*） [6] 以及采用按工作量排序的搜索的 LazyPRM* 版本。

将所有碰撞对象分为*i)* 机器人、*ii)* 可移动物体和*iii)* 静态环境后，场景的完整碰撞检测可拆分为五项检测（图1）：机器人-机器人、机器人-静态环境、机器人-可移动物体、可移动物体-可移动物体，以及静态环境-可移动物体。在这些检测中，前两项（机器人-机器人和机器人-静态环境）可在所有可能的运动规划查询中重复利用，前提是机器人和静态环境保持不变。这种信息分解与重复利用不会改变所用运动规划算法的完备性或最优性。

我们在多个模拟机器人问题上展示了我们的方法。我们提供了消融研究，并比较了

采用我们方法的规划器与其他不重用信息的最先进单次查询规划器。

最后，我们还展示了在需要反复解决相似操作规划问题的场景中（即多查询操作规划场景）使用的规划器。我们的方法既能在单个操作规划问题中的不同动作之间，也能在多个操作查询之间复用已获得的信息，而且只需对规划算法进行最小限度的修改。

II. 相关工作

A. 操作规划中的路径规划

操作规划是任务与运动规划（TAMP）的一种特殊情况，旨在寻找一个动作序列（可能还包括其参数化形式）及相应的路径，以便将一组物体操纵至期望的最终状态。在操作规划过程中，环境可能会发生变化，例如移动物体。此类问题常见于建造 [7], 或装配规划 [8]。

概率路线图树（PTR） [4]随机采样动作，然后尝试在与该动作对应的模式中寻找路径。PTR 使用的路径规划器是单查询规划器，会针对每个动作和每次尝试重新实例化，且不会复用任何信息。

[9], 中介绍了操作规划问题的一种特殊情况，其中作出了一个强假设：转换不会改变自由构型空间。虽然这将找到可行解所需的尝试次数降至最低，但不同动作之间不会重用任何信息。

可以重用以往规划查询中获得的信息，引导新的路径偏向那些已知在先前规划查询中有效的路径 [5], [10], [11]。然而，尽管路径会偏向已知有效的路径，以前进行过的（可能可重用的）碰撞检测并未被重用。

Schmitt 等人 [12]提出了一种操作规划方法，通过对模式的参数化进行采样，并为每种参数化构建单独的路线图。随后，该方法尝试连接这些独立的路线图。这种方法可以重用先前的规划结果，但仅限于一种非常特殊的情况，而且并非有意为之。

任务与运动引导树 [13] 采用非对称搜索来推迟计算开销大的操作，但不会明确重用各种求解尝试中获得的信息，也不会重用多模态规划问题中不同模式之间的信息。

上述方法都没有在其路径规划子程序中重用验证工作，而我们提出的方法可以做到这一点。我们提出的方法同样可以应用于上述方法，使其也能够重用此前投入的工作量。

B. 多查询路径规划

多查询路径规划旨在高效地解决同一（静态）空间中的多个运动规划查询。许多多查询规划器都基于概率路线图（PRM） [14],及其变体（例如，（Lazy）PRM(*) [1],[15]）。如果某条边出现在此前已验证过的路径中，这些规划器会重用先前投入的规划工作量，但不会主动尝试最大限度地重用。在实践中，这并不会在一次规划查询中带来多少重用。

Lightning [16]是一个基于稀疏路网（SPARS） [17]的框架，可从数据库中检索现有规划方案，并根据当前环境对其进行调整。由于向 SPARS 中插入路径以及从中检索路径的计算开销可能很大，因此总体规划器可能比不重用信息的规划器更慢。

经验图（E-图） [18]会主动重用此前完成碰撞检测的边，但由于需要反复运行全点对最短路径算法，其计算速度较慢。

可重构随机森林（RRF） [19] 将图搜索与基于树的方法相结合，以构建多查询规划器。通过分别在起点和目标处锚定并生长新树，并朝向其他现有树规划路径，该方法在多次查询过程中迭代构建路网。该方法不提供任何最优性保证，也不是随时可用的。此外，所构建的路网可能无限增长，从而减慢后续规划查询的速度。

EIRM* [6], 通过按验证工作量排序搜索，积极重用先前投入的计算工作量。它采用双向搜索，首先为后续的底层图搜索计算启发式信息。与其他多查询规划器相比，EIRM* 会积极管理多个查询过程中产生的图增长。在初始解时间方面，EIRM* 的表现优于其他多查询规划器，最高可快一个数量级。

这些方法都无法直接应用于操作规划问题，因为它们假设环境是静态的。通过分解碰撞检测，从而实现验证工作量的重用，便可在多个动作之间重用验证工作量。

III. 将操作规划视为多查询问题

A. 问题陈述与符号表示

我们考虑一个由 N 碰撞对象组成的场景，这些对象的位姿属于 $p \in \mathrm{SE(3)}$。我们用 $\mathcal{O} \subseteq \{1, ..., N\}$ 表示可移动物体，并用 $\tau \in \mathcal{O}$ 指代特定物体。随后，我们用 $\theta_\tau \in \mathrm{SE(3)}$ 表示物体 τ 相对于其父坐标系的变换；该父坐标系可以是机器人夹爪（如果物体当前正在被操作），也可以是静态环境（如果物体处于静止状态）。

在操作规划中，会执行一系列动作，例如“抓取(obj1) - 放置(obj1)”[1]。我们将 $\Theta_i = \{\theta_\tau | \tau \in \mathcal{O}\}_i$ 称为场景在第 i 个动作期间的 参数化 。我们用 $\mathcal{M}_i \subseteq \mathcal{O}$ 表示该动作序列中第 i 个动作期间被操作（例如移动）的物体集合（在前面的抓取和放置

[1]关于操作规划中动作这一概念的详细讨论，请参阅例如 [20]。

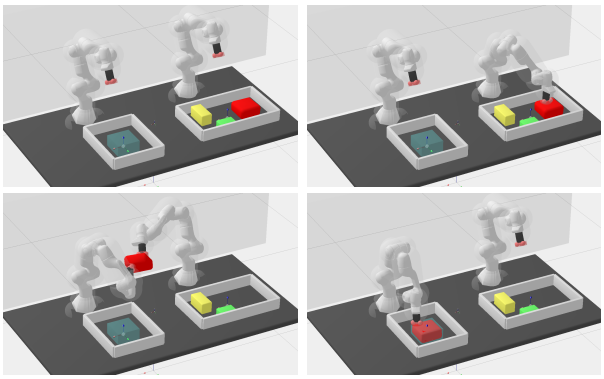


图2：交接序列中各动作的起始和目标构型。在此示例中，每个动作中被操作的物体为 $\mathcal{M}_1 = \{\}$，$\mathcal{M}_2 = \mathcal{M}_3 = \{\text{obj}_1\}$；$\theta_1$分别表示红色物体相对于桌面的变换，以及相对于机器人手臂夹爪的变换。

示例中，$\mathcal{M}_1 = \{\}$，$\mathcal{M}_2 = \{\text{obj}1\}$），并用 $\theta_{\mathcal{M}_i}$ 表示参数化 Θ_i 中对应于移动物体的子集。

我们用 $\mathcal{C} \subseteq \mathbb{R}^n$ 表示机器人的构型空间，用 $q \in \mathcal{C}$ 表示机器人的特定构型[2]。动作i 期间的无碰撞构型空间为 $\mathcal{C}_{\text{free},\ i}(\Theta_i)$，它取决于我们当前正在规划的动作 i 以及所有可移动物体的参数化。机器人在动作 i 中的起始状态为$q_{i,\ \text{start}}$，目标状态为 $q_{i,\ \text{goal}}$。虽然这种方法很容易扩展到在起始状态集合与目标状态集合之间进行规划，但为简化记号，这里仅介绍起始状态和目标状态均为单一状态的版本。

给定一个元组 $\alpha_i = (q_{i,\ \text{start}},\ q_{i,\ \text{goal}},\ \mathcal{M}_i,\ \Theta_i)$，其中包含起始状态、目标状态、移动物体的子集以及可移动物体的参数化，目标是找到一条无碰撞路径 $\pi_i(s)$: $[0,1] \mapsto \mathcal{C}_{\text{free},\ i}$，满足 $\pi_i(0) = q_{i,\ \text{start}}$，$\pi_i(1) = q_{i,\ \text{goal}}$。操作规划问题中的路径规划由一系列$\mathcal{A} = \{\alpha_1, ..., \alpha_n\}$路径规划问题组成，其中，目标状态与前一动作的起始状态重合，并且第 i个动作结束时的无碰撞环境等于第$(i+1)$个动作开始时的环境。图2展示了这样一个序列中起始状态和目标状态的示例。在操作规划中计算抓取位置本身就是一个开放性研究问题，我们假设这些位置已知。

我们用 $e(q,\ q')$ 表示状态 q 和 q' 之间的规划工作量。状态之间规划工作量的一个例子是：检查两个状态之间的线性连接是否无碰撞所需的计算开销。

我们用 $\xleftarrow{+}$ 表示两个集合的并，即 $A \xleftarrow{+} B$ 表示 $A \leftarrow A \cup B$。

在某些情况下，可能希望最小化路径上的代价$c(\pi)$。在本文中，我们使用路径长度作为

[2]如果场景中有多个机器人，我们假设 $\mathcal{C} = \mathcal{C}_1 \times ... \times \mathcal{C}_n$，即构型空间是每个机器人的构型空间的堆叠。

算法1： 有效性检查(q, $\text{check}_{\text{r/s}}$, $\text{check}_{\text{r/o}}$, $\text{check}_{\text{o/s}}$)

$\mathcal{C}$.设置配置(q) $\text{coll}_{\text{r/s}} \leftarrow$false; $\text{coll}_{\text{r/o}} \leftarrow$false; $\text{coll}_{\text{o/s}} \leftarrow$false **如果** $\text{check}_{\text{r/s}}$ = $\mathcal{C}$ ∨ .机器人-静态对象碰撞() 。 $\text{coll}_{\text{r/o}}$ = $\mathcal{C}$ ∨ .机器人-可移动对象碰撞() 。 $\text{coll}_{\text{o/s}}$ = $\mathcal{C}$.可移动物体-静态物体碰撞() **返回**($\text{coll}_{\text{r/s}}$, $\text{coll}_{\text{r/o}}$, $\text{coll}_{\text{o/s}}$)

算法2： 验证(path, Θ_i, $\mathcal{M}_i$)

对于 $l \in$ path.edges 执行 $\text{check}_{\text{r/s}} \leftarrow l$.validity == 未知 $\text{check}_{\text{o/s}} \leftarrow l$.validity$[\theta_{\mathcal{M}_i}]$ == 未知 $\text{check}_{\text{r/o}} \leftarrow l$.validity$[\Theta_i]$ == 未知 对于 $q_{\text{interp}} \in$ 插值(l, 分辨率) 执行 (碰撞$_{\text{r/s}}$, 碰撞$_{\text{r/o}}$, 碰撞$_{\text{o/s}}$) ←有效性检查(q_{interp}, $\text{check}_{\text{r/s}}$, $\text{check}_{\text{r/o}}$, $\text{check}_{\text{o/s}}$) 如果存在任一碰撞$_{\text{r/s}}$, 碰撞$_{\text{r/o}}$, 碰撞$_{\text{o/s}}$ 如果碰撞$_{\text{r/s}}$ l.validity ← 无效 如果碰撞$_{\text{o/s}}$ l.validity$[\theta_{\mathcal{M}_i}]$ ← 无效 如果碰撞$_{\text{r/o}}$ l.validity$[\Theta_i]$ ← 无效 返回 false **返回**真

代价，但所提出的方法适用于任意加性代价。我们将经过状态 q、使用 $f(q)$ 的路径的代价可容许估计记为。

B. 分解式碰撞检测

操作规划问题中动作 i 下的无碰撞构型空间可表示为三个无碰撞构型空间的交集：

$$\mathcal{C}_{\text{free},i} = \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{free}}(\theta_{\mathcal{M}_i}) \cap \mathcal{C}_{\text{free}}(\Theta_i), \tag{1}$$

其中，第一部分与环境的参数化无关，第二部分仅取决于当前动作中移动的物体，第三部分则取决于所有可移动障碍物。

采用前文介绍的分组方式（图1），我们得到 5 项碰撞检测，可映射到分解后的构型空间（式（1））：

1) $\mathcal{C}_{\text{free}}$：机器人-机器人、机器人-静态物体 2) $\mathcal{C}_{\text{free}}$（$\theta_{\mathcal{M}_i}$）：可移动物体-静态物体 3) $\mathcal{C}_{\text{free}}$（$\Theta_i$）：可移动物体-可移动物体、机器人-可移动物体。

由于 1）独立于动作 i 和可移动物体的参数化，因此碰撞检测结果可以在所有规划查询中重复使用。如果参数化 Θ_i 与之前的碰撞检测相同，则第 2）部分可以重复使用；如果当前动作 $\mathcal{M}_i$ 中的移动物体集合与之前碰撞检测中的相同，则第 3）部分可以重复使用。

算法 3： 操作规划中的路径规划。

路径 $\leftarrow \emptyset$
对 $(q_{i,\text{start}}, q_{i,\text{goal}}, \mathcal{M}_i, \Theta_i) \in \mathcal{A}$ **执行**
  $\mathcal{C}$.更新参数化$(\mathcal{M}_i, \Theta_i)$
  路径 $\xleftarrow{+}$规划$(q_{i,\text{start}}, q_{i,\text{goal}}, \mathcal{M}_i, \Theta_i)$
返回路径

C. 算法

算法 1 和算法 2 展示了如何重复利用碰撞检测的已有信息：对边 l 进行碰撞检测时，分解式碰撞检测各部分的有效性状态可以是未知、有效或无效。如果任一部分的有效性状态为无效，则该边无效，因此不会出现在规划算法的图搜索所找到的路径中。如果某部分已知有效，则无需对该部分进行碰撞检测。因此，只有当某部分的状态未知时，才需要检查其有效性（算法 2，第 2 至 4 行；算法 1，第 3、5 和 7 行）。随后，通过在给定分辨率下对边上的离散点进行插值来对边进行碰撞检测（算法 2，第 5 行），并检查插值得到的配置是否有效（算法 2，第 6 行）。边完成碰撞检测且未发现碰撞后，其所有有效性分量都会被标记为有效（算法 2，第 12 至 14 行）。如果边发生碰撞，则将发生碰撞的部分标记为无效（算法 2，第 8 至 10 行），并由 validate 返回 false（算法 2，第 11 行）。

算法 3 展示了求解操作规划问题时如何实现这种复用。我们在两种能够有意复用工作量的路径规划算法中演示分解式碰撞检测：按工作量排序的 LazyPRM*（算法 4）和 EIRM*（算法 5）。在这两种规划算法中，我们假设使用边隐式随机几何图 X_{RGG} 来存储当前构成自由空间近似的样本。

a) 按工作量排序的 *Lazy PRM**（*eo-LazyPRM**）：我们实现了一种基于 PRM* 且采用稀疏碰撞检测的算法，类似于 [21]，；也就是说，在图搜索过程中，对边进行稀疏检测。找到候选路径后，再对其进行密集检测（使用算法 2）。初始时，我们按工作量排序搜索，并以代价作为平局判定标准。为实现最优性，在找到初始解后，搜索改为按代价排序，并以工作量作为平局判定标准。

我们还实现了 EIRM* 中采用的图规模管理方法：使用批次回溯：所有采样状态都存储在缓冲区 $\mathcal{B}$ 中，一旦调用细化近似（算法 6），就会从缓冲区向当前图中添加一批新样本（大小为 m）。在运动规划查询开始时，我们从空图开始，随后将缓冲区中的样本 $\mathcal{B}$ 重新添加回图中。更多详情请参阅 [6]。

b) 努力知情路线图（*EIRM**）：EIRM* 采用非对称双向搜索，其中计算开销低廉的反向搜索用于计算针对特定问题和近似的启发式信息。这些启发式——

算法 4： 有序 有序 Lazy PRM*: 规划$(q_{i,\text{start}}, q_{i,\text{goal}}, \Theta_i, \mathcal{M}_i)$。更改 相比 to LazyPRM* 橙色。

$X_{\text{RGG}} \leftarrow \{q_{i,\text{start}}, q_{i,\text{goal}}\}$；$c_{\text{best}} \leftarrow \infty$；路径 $\leftarrow \emptyset$；$i_{\text{buffer}} \leftarrow 0$
当尚未停止时 **执行**
  $X_{\text{RGG}} \xleftarrow{+}$细化近似（批次大小，$c_{\text{best}}$）
  如果 c_{best} **是** ∞
    $p \leftarrow X_{\text{RGG}}$.按搜索工作量排序()
  else
    $p \leftarrow X_{\text{RGG}}$.按搜索代价排序()
  如果 验证$(p, \Theta_i, \mathcal{M}_i)$ **且** 代价$(p) < c_{\text{best}}$
    $c_{\text{best}} \leftarrow$ 代价(p)
    路径 $\leftarrow p$
返回路径

算法 5： EIRM*: plan $(q_{i,\text{start}}, q_{i,\text{goal}}, \Theta_i, \mathcal{M}_i)$

$X_{\text{RGG}} \leftarrow \{q_{i,\text{start}}, q_{i,\text{goal}}\}$；$c_{\text{best}} \leftarrow \infty$；路径 $\leftarrow \emptyset$；
$i_{\text{buffer}} \leftarrow 0$ **当**未停止时**执行** **如果**最佳反向边能改进 sol() $\bar{X}_{\text{RGG}}$。反向搜索() **否则如果**最佳正向边能改进 sol() 路径 $\Leftarrow X_{\text{RGG}}$。正向搜索() **否则**
  $X_{\text{RGG}} \xleftarrow{+}$细化近似（批次大小，$c_{\text{best}}$） **返回**路径

随后使用这些统计信息来引导正向搜索，并对各条边进行完整的碰撞检测。与 [6] 中所述实现相比，核心变化在于碰撞检测方式以及构型采样方式（即分别使用算法2和算法6进行有效性检查和细化图近似）。

1) 努力估计：我们使用验证工作量作为规划工作量 e的估计，即验证一条边无碰撞所需的计算开销。两个状态之间的规划工作量 e 等于它们之间的欧几里得距离除以碰撞检测分辨率。

鉴于碰撞检测可以分解为三个可部分复用的独立部分，我们也可以将验证工作量分解为三部分。我们通过累加三个碰撞检测各自所需的相对工作量估计值，来估算一条边所需的验证工作量，即：

$$e(q, q') = e_{r/s}(q, q') + e_{o/s}(q, q'|\theta_{\mathcal{M}_i}) + e_{r/o}(q, q'|\Theta_i), \quad (2)$$

其中，$e_{r/s}$ 对应于检查一条边是否位于 $\mathcal{C}_{\text{free}}$中所需的时间，$e_{o/s}$ 对应于 $\mathcal{C}_{\text{free}}(\theta_{\mathcal{M}_i})$，而$e_{r/o}$ 对应于 $\mathcal{C}_{\text{free}}(\Theta_i)$。如果 q，q'之间的边此前已检查过，因此已标注其有效性，那么总和中的这一部分为零。

EIRM* 需要估计先验的剩余工作量d （即从目标状态到目标状态的工作量），并利用该估计对其反向搜索进行排序。与 [6],

算法 6： 细化近似(m, c_{best})。
与原始版本相比的更改 [6] 以 橙色标出。

$\mathcal{B} \equiv (b_1, b_2, \ldots, b_n)$
$M \leftarrow \emptyset$
当 $|M| \leq m$ **时执行**
  如果 $i_{\text{buffer}} > |\mathcal{B}|$
    $q \leftarrow$均匀采样()
    如果有效性检查(q, true, false, false)
      $\mathcal{B} \xleftarrow{+} q$
  如果 $f(b_{i_{\text{缓冲区}}}) < c_{\text{best}}$ **并且** 有效性检查(q, false, true, true)
    $M \xleftarrow{+} b_{i_{\text{buffer}}}$
  $i_{\text{buffer}} \leftarrow i_{\text{buffer}} + 1$
返回 M

其中 d为 0，我们使用信息量更丰富的

$$d(q) = \max\{e(q, q_{i,\text{goal}}) - (e_{\text{reusable}} - e_{\text{claimed}}), 0\}. \quad (3)$$

我们会跟踪迄今投入的可重复利用工作量总和 e_{reusable}，以及迄今在搜索过程中认领的工作量 e_{claimed}（即已重复利用的工作量）。因此，剩余工作量是验证当前状态与目标之间的边所需的工作量 $e(q, q_{i,\text{goal}})$ 与剩余工作量之间的差值。也就是说，我们乐观地假设所有剩余工作量都属于通往目标的路径。

IV. 实验

我们在图 3 所示的场景中评估[3] 我们的方法。动作序列均为抓取与放置序列，交接场景除外：在该场景中，右臂抓取物体并将其交给左臂，随后由左臂将物体放置在目标位置。

在以下场景中，我们假设操作问题的抓取位置已知。我们依次计算操作规划问题中的运动，即先求解第一个动作的路径规划问题，再求解第二个动作的路径规划问题，依此类推。

我们使用快速碰撞库 (FCL) [23] 进行碰撞检测。FCL 原生支持对碰撞对象进行分组。通过分解碰撞检测，并避免检测机器人-机器人和机器人-静态环境之间的交互，节省的相对时间为 20% 至 50%。

A. 算法与参数

我们将采用我们的方法的规划器与 OMPL [24] 中的 RRT-Connect 和 EIT* [21] 实现进行比较，后两者作为基线，不会在不同查询之间复用任何信息。此外，我们还展示了一个使用分解式碰撞检测的 LazyPRM* 版本，即能够复用信息，但按代价对搜索进行排序，因此不会主动复用信息。为了控制该规划器的图增长，我们采用与按工作量排序的版本相同的方法。

我们为这些问题选择碰撞检测分辨率，使假阳性率低于 1‰，也就是说，每千条边中，约有一条被误认为有效，实际上却无效。因此，抽象问题的碰撞检测分辨率约为 10^{-4}，机器人问题的碰撞检测分辨率约为 10^{-3}。对于 EIT*、EIRM* 和 LazyPRM* 变体，我们使用 500 个样本的批次大小。

B. 初始解时间与成本

每个规划器使用不同的伪随机种子运行 100 次。在这 100 次运行期间，起点和目标始终保持不变。由于我们关注的是规划器找到初始解所需的时间，因此，当针对当前路径规划问题找到解后，规划器即停止运行，并开始操作规划序列中的下一个路径规划问题。

表 I 展示了找到初始解的中位时间及相应的初始成本中位数。成功率，即在规定时间内为完整操作规划问题中的所有路径找到解的规划器所占的百分比，如图 4 所示。

在我们的所有测试中，复用信息的规划器在初始解时间方面均优于不复用信息的规划器。复用信息的规划器所得到的初始解代价往往略低于不复用信息的规划器所得到的初始解代价。

C. 成本演化

为分析规划器的成本收敛情况，我们针对操作规划问题中的每个规划查询，使用不同的伪随机种子，让每个规划器在 10 秒内运行 100 次。

图5展示了每个规划器的解随时间变化的成本演化，以及相应的成功率图。在第二个动作中可以明显看出速度差异，多查询规划器可以在此处复用信息。

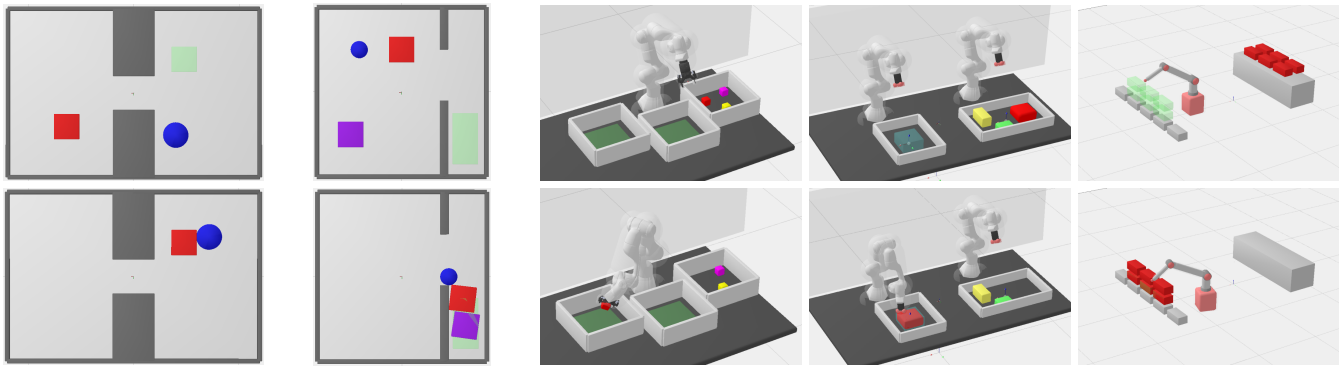
复用信息的规划器比 EIT* 更慢地收敛到最优解。由于需要用均匀分布样本填充缓冲区，以应对未来的查询，因此我们无法直接对知情集进行采样。因此，为了对知情集进行采样，我们采用拒绝采样；当逐渐接近最优解时，这种方法的计算开销高于直接对知情集进行采样。

D. 多查询操作规划

我们展示了一个示例，在该示例中，我们解决了同一系列的多个问题，即同一问题的不同变体，例如物体的尺寸不同，或初始位置和最终位置不同。这类问题经常出现在例如建造 [8]或装配规划 [7]中。对于与可移动障碍物相关的碰撞信息复用，还需要格外谨慎：如果它们的形状发生变化，则无法复用与其相关的碰撞信息。因此，在本实验中，仅复用了静态环境和机器人的信息。

我们从同一问题族中创建了 10 个实例（即初始目标和最终目标，以及可移动对象的形状各不相同的场景），如图6所示。每个

[3]所有实验均在一台配备 Intel i7-8565U CPU @ 1.80GHz 处理器和 16GB 内存的笔记本电脑上运行。我们采用了 [22] 中介绍的基准测试最佳实践。

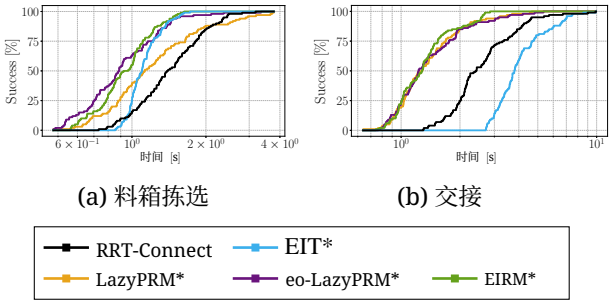


(a) 墙隙　(b) 多重插入　(c) 料箱拣选　(d) 交接　(e) 砖墙

图 3：基准测试中使用的场景图像。起始配置显示在上方，目标配置显示在下方。

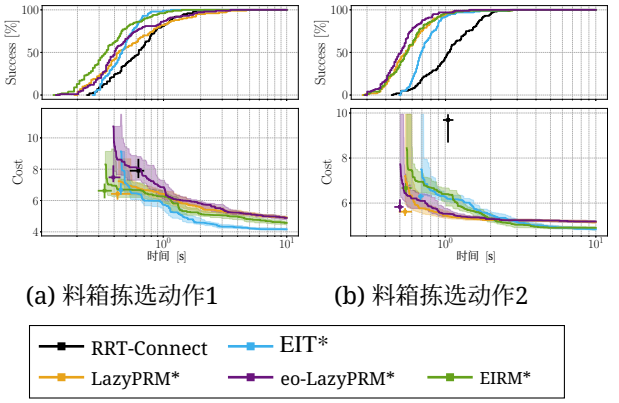
表 I：100 次运行中获得初始解的中位时间及其对应的初始代价。下标和上标分别表示非参数 95% 置信界限的下界和上界。每个问题的最佳值以**粗体**标出。虚线以上的规划器是单查询规划器，虚线以下的规划器是采用分解式碰撞检测的多查询规划器。

场景 动作数量	墙隙（二维） 2		多重插入（三维） 4		抓取分拣（七维） 2		砖墙（8D） 12		交接（14D） 3	
	t_{init}	c_{init}	t_{init}	c_{init}	t_{init}	c_{init}	t_{init}	c_{init}	t_{init}	c_{init}
RRT-Connect	$0.387^{+0.04}_{-0.04}$	$6.31^{+0.34}_{-0.34}$	$0.728^{+0.08}_{-0.04}$	$28.26^{+1.27}_{-1.21}$	$1.40^{+0.15}_{-0.10}$	$17.22^{+0.90}_{-0.65}$	$5.48^{+0.37}_{-0.53}$	$141.37^{+4.98}_{-4.27}$	$2.42^{+0.30}_{-0.24}$	$23.39^{+0.43}_{-0.65}$
EIT*	$0.100^{+0.00}_{-0.00}$	$3.37^{+0.01}_{-0.01}$	$0.127^{+0.01}_{-0.00}$	$10.71^{+0.11}_{-0.40}$	$1.10^{+0.03}_{-0.05}$	$12.98^{+0.49}_{-0.44}$	$1.73^{+0.06}_{-0.05}$	$58.59^{+1.14}_{-2.47}$	$3.80^{+0.22}_{-0.18}$	$15.63^{+0.97}_{-0.40}$
LazyPRM*	$0.206^{+0.01}_{-0.02}$	$\mathbf{3.34}^{+0.01}_{-0.01}$	$0.178^{+0.01}_{-0.01}$	$\mathbf{9.91}^{+0.13}_{-0.07}$	$1.13^{+0.15}_{-0.10}$	$\mathbf{11.84}^{+0.48}_{-0.27}$	$2.49^{+0.36}_{-0.23}$	$\mathbf{48.36}^{+1.50}_{-1.59}$	$1.27^{+0.11}_{-0.13}$	$\mathbf{11.09}^{+0.26}_{-0.22}$
eo-LazyPRM*	$0.150^{+0.01}_{-0.01}$	$3.47^{+0.03}_{-0.01}$	$0.163^{+0.01}_{-0.01}$	$11.94^{+0.15}_{-0.35}$	$\mathbf{0.884}^{+0.04}_{-0.05}$	$13.47^{+1.13}_{-0.57}$	$1.86^{+0.14}_{-0.13}$	$55.34^{+2.92}_{-2.36}$	$\mathbf{1.23}^{+0.13}_{-0.06}$	$11.19^{+0.94}_{-0.20}$
EIRM*	$\mathbf{0.098}^{+0.00}_{-0.00}$	$3.42^{+0.02}_{-0.02}$	$\mathbf{0.114}^{+0.01}_{-0.00}$	$10.78^{+0.33}_{-0.14}$	$0.945^{+0.07}_{-0.09}$	$13.93^{+1.15}_{-0.46}$	$\mathbf{1.55}^{+0.04}_{-0.06}$	$51.25^{+0.66}_{-1.85}$	$1.28^{+0.06}_{-0.14}$	$11.47^{+0.64}_{-0.33}$



(a) 料箱拣选　(b) 交接

图 4：完整操作规划问题中，各规划器随时间变化的成功运行次数百分比。



(a) 料箱拣选动作1　(b) 料箱拣选动作2

图5：抓取分拣两种模式的成本演化图。成本演化图（下方）以粗线表示给定时间的成本中位数，阴影区域表示非参数95%置信区间。方块表示查询的初始解时间中位数及对应的初始代价中位数。未找到解时，代价视为无穷大。

随后，让规划器在相同的 100 个序列上运行，这些序列由打乱 10 个操作规划问题实例的顺序构成。

图 7 展示了每次查询的初始解时间中位数。采用分解式碰撞检测的规划器表现优于单查询规划器，优势约为 2.5 倍。在多查询规划器中，主动复用先前投入的工作量的规划器优于未采用此方法的规划器。

E. 消融实验

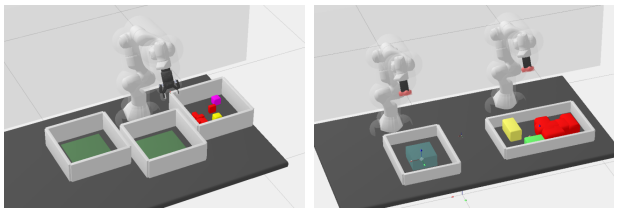
1) 图规模管理：我们将使用按工作量排序的搜索的算法与不使用该方法的版本进行比较
批次回溯，即从用于寻找前一个解的所有有效样本开始。

图8展示了

对于未采用图规模管理的算法，在第 IV-D 节所讨论的相同实验中的初始解时间中位数。

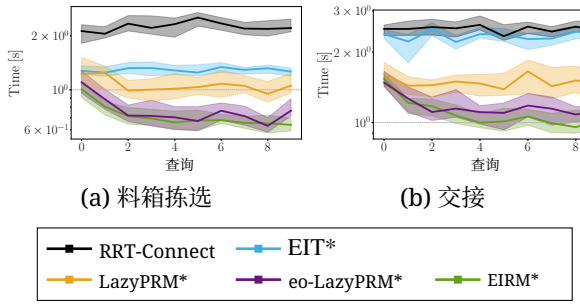
左侧场景中的性能相当，但在交接场景中，未管理版本的表现明显更差。

请注意，在此处的实验中，这些路径规划运行



(a) 料箱拣选　　(b) 交接

图6：问题族中一些具有不同物体尺寸和位置的实例示意图。



(a) 料箱拣选　　(b) 交接

图7：多个操作规划查询的运动规划时间中位数。粗线表示 100 次运行的规划时间中位数，阴影区域表示非参数95%置信区间。

在找到初始解后便停止了。如果规划器在找到初始解后继续运行以优化路径，那么这里所见的效果会更明显；甚至可能会出现这种情况：如果不主动进行图规模管理，由于图的密度过高，在给定时间内将无法再找到解（如[6]中所述）。

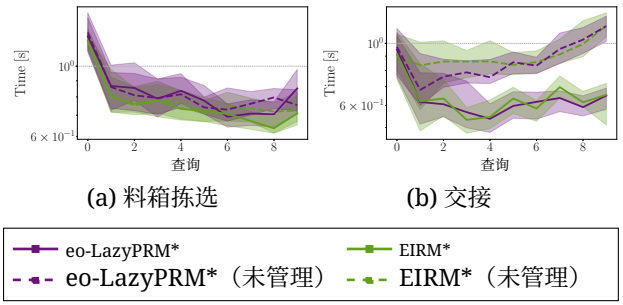
2）主动与被动的计算工作复用：我们将一种使用分解式碰撞检测、但采用按代价排序的搜索的规划器与其他方法进行比较；这也是 LazyPRM* 中的标准做法。因此，尽管该规划器能够复用信息，却不会主动尝试这样做。从表 I 和图7中都可以看出，按工作量排序的搜索所得的解所需时间少于按代价排序的搜索。不过，按代价排序的搜索得到的代价更低。

3）碰撞检测分辨率的影响：我们在图 9 中比较了两个具有不同碰撞检测分辨率的场景所需的规划时间。RRT-Connect 的耗时大致随碰撞检测分辨率线性增长，而基于搜索的规划器表现出更好的扩展性。

请注意，在这些设置中，碰撞检测分辨率大于 10^{-3} 会导致获得无效路径的概率高得难以接受。

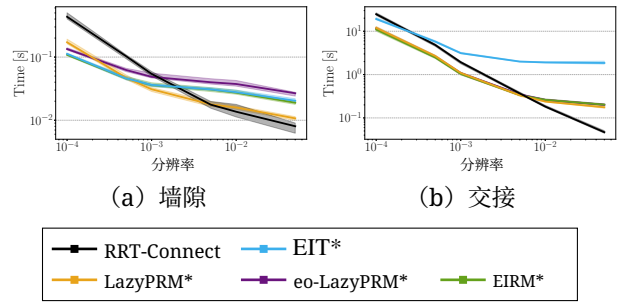
V. 讨论与局限性

在初始解时间方面，会在多次规划查询中重复利用碰撞信息的规划器始终优于不重复利用信息的规划器，且解的质量相当或更好（表 I）。与不重复利用信息的规划器相比，差异



(a) 料箱拣选　　(b) 交接

图 8：比较使用批次回溯的规划器与不使用该功能的版本，测试场景与第 IV-D 节相同。粗线表示 100 次运行的规划时间中位数，阴影区域表示非参数 95% 置信区间。



（a）墙隙　　（b）交接

图 9：在不同碰撞检测分辨率下，运行 100 次时找到初始解的中位时间。阴影区域表示 95% 的非参数置信区间。

在同一问题需要求解多次时最为显著，例如任务与运动规划问题，或通过将问题分解为较小子问题来求解的长时域问题（图7）。如果预计会针对同一或类似问题进行多次查询，且路径规划问题并非简单问题，那么分解问题并重复利用碰撞信息的方法会更加有用。

如第 IV-E.3 节的碰撞检测分辨率消融实验所示，本文提出的方法在碰撞检测相对耗时的场景中改进最为明显，这通常也是实际机器人学应用中的情况。虽然在简单场景中使用该方法也有优势，但优势并不那么显著。

根据 [3] 中的分析，当路径得到优化且样本数量增加时，最近邻计算开始占据规划时间的主要部分。要进一步缩短这部分规划时间，一种可行的方法是实现最近邻计算缓存。

我们认为，正如文献 [6]中针对 EIRM* 已经讨论的那样，对何时以及如何将有效边提升至更早的批次进行严谨分析，将有助于改进所提出的方法。一种可能的实现方式或许类似于 [25]，即对构型空间的核心特征进行在线学习，并将这些特征提升至第一批采样点。

VI. 结论

我们提出了一种供算法在序列操作规划中复用信息的方法：将碰撞检测分解为对环境变化不敏感的信息，以及对环境变化敏感的部分。这样，便能够在操作规划中出现多个规划查询时，高效复用先前计算的信息，从而使路径规划时间较当前最先进方法最多快两倍。

尽管本文以碰撞检测为例，介绍了基于分解的方法来复用计算工作，但其背后的思想也可以更广泛地应用于运动有效性检查和状态有效性检查。例如，力或力矩约束也可以分解为相互独立的部分，以及依赖于场景特定参数化的部分。

未来，我们希望将这种方法应用于任务与运动规划场景。在这些场景中，我们提出的这类复用会在规划过程中自然出现：在一个动作序列中测试多个高级规划，并评估多个不同的抓取位置。我们还认为，这种方法适用于分解式多机器人路径规划 [26]。

本文仅关注运动规划部分，动作序列和抓取位置均为给定。未来的工作中，可以利用运动规划查询过程中获得的信息，例如，通过倾向于此前已验证可行的抓取方式，来帮助计算有效的抓取位置。

参考文献

[1] K. Hauser，《渐近最优运动规划中的惰性碰撞检测》，载于*IEEE*机器人与自动化国际会议（*ICRA*）论文集。IEEE，2015年，第2951–2957页。[2] G. S´anchez 和 J.-C. Latombe，《采用惰性碰撞检测的单查询双向概率路线图规划器》，载于机器人学研究。Springer，2003年，第403–417页。[3] M. Kleinbort、O. Salzman 和 D. Halperin，碰撞检测还是最近邻搜索？论基于采样的运动规划中的计算瓶颈，2020年，第624–639页。[4] K. Hauser，《利用连续动作和非确定性运动规划查询进行任务规划》，载于*AAAI*任务规划与运动规划衔接研讨会论文集，2010年。[5] Z. Kingston、C. Chamzas 和 L. E. Kavraki，《利用经验改进多模态问题中叶层上的约束规划》，载于*IEEE/RSJ*智能机器人与系统国际会议（*IROS*）论文集，2021年，第6922–6927页。[6] V. N. Hartmann、M. P. Strub、M. Toussaint 和 J. D. Gammell，《努力知情路线图（EIRM*）：通过主动复用验证工作量实现高效的渐近最优多查询规划》，载于国际机器人学研究研讨会，2022年，arXiv:2205.08480 [cs.RO]。[7] V. N. Hartmann、A. Orthey、D. Driess、O. S. Oguz 和 M. Toussaint，《用于建筑装配的长时域多机器人重排规划》，*IEEE*机器人学汇刊，2022年。[8] N. Funk、G. Chalvatzaki、B. Belousov 和 J. Peters，《通过结构化表示和搜索实现用于机器人建筑施工的Learn2assemble》，载于机器人学习会议（*CoRL*），2021年。[9] P. Englert、I. M. R. Fern´andez、R. K. Ramachandran 和 G. S. Sukhatme，《序列流形上的基于采样的运动规划》，载于机器人学：科学与系统（*R:SS*）论文集，2021年。[10] T. Lai 和 F. Ramos，《Ltr*：利用惰性经验图快速重新规划连续执行的任务》，载于*IEEE/RSJ*智能机器人与系统国际会议（*IROS*）论文集，2022年，第8784–8790页。[11] G. Zuo、M. Li、J. Yu、C. Wu 和 G. Huang，《用于重复抓取和放置的惰性示范图高效运动规划方法》，仿生学，第7卷，第4期，2022年。[12] P. S. Schmitt、W. Neubauer、W. Feiten、K. M. Wurm、G. V. Wichert 和 W. Burgard，《最优的基于采样的操作规划》，载于*IEEE*机器人与自动化国际会议（*ICRA*）论文集。IEEE，2017年，第3426–3432页。[13] W. Thomason、M. P. Strub 和 J. D. Gammell，《任务与运动引导树（TMIT*）：几乎必然渐近最优的集成任务与运动规划》，*IEEE*机器人与自动化快报，第7卷，第4期，第11 370–11 377页，2022年。[14] L. E. Kavraki、P. Svestka、J.-C. Latombe 和 M. H. Overmars，《高维构型空间中的路径规划概率路线图》，*IEEE*机器人与自动化国际会议论文集（*ICRA*），第12卷，第4期，第566–580页，1996年。[15] S. Karaman 和 E. Frazzoli，《最优运动规划的基于采样的算法》，《国际机器人研究杂志》，第30卷，第7期，第846–894页，2011年。[16] D. Berenson、P. Abbeel 和 K. Goldberg，《一种能够从经验中学习的机器人路径规划框架》，载于*IEEE*机器人与自动化国际会议（*ICRA*）论文集。IEEE，2012年，第3671–3678页。[17] A. Dobson 和 K. E. Bekris，《用于渐近近似最优运动规划的稀疏路线图伸展器》，《国际机器人研究杂志》，第33卷，第1期，第18–47页，2014年。[18] M. Phillips、B. J. Cohen、S. Chitta 和 M. Likhachev，《E-图：利用经验图为规划奠定基础》，载于机器人学：科学与系统（*R:SS*）论文集，第5卷，2012年，第110页。[19] T.-Y. Li 和 Y.-C. Shie，《一种结合路线图管理的运动规划增量学习方法》，载于*2002*年*IEEE*国际机器人与自动化会议论文集（目录号*02CH37292*），第4卷。IEEE，2002年，第3411–3416页。[20] M. Toussaint，《逻辑几何规划：一种结合任务与运动规划的基于优化的方法》，载于国际人工智能联合会议（*IJCAI*）论文集，2015年。[21] M. P. Strub 和 J. D. Gammell，《自适应信息树（AIT*）与努力信息树（EIT*）：非对称双向基于采样的路径规划》，《国际机器人研究杂志》，第41卷，第4期，第390–417页，2022年4月。[22] J. D. Gammell、M. P. Strub 和 V. N. Hartmann，《规划器开发工具（PDT）：用于开发和测试运动规划器的可复现实验与统计分析》，载于运动规划性能评估研讨会（*EMPP*）论文集，*IEEE/RSJ*智能机器人与系统国际会议（*IROS*），日本京都，2022年10月23日。[23] J. Pan、S. Chitta 和 D. Manocha，《FCL：用于碰撞与邻近查询的通用库》，载于*IEEE*机器人与自动化国际会议（*ICRA*）论文集，2012年，第3859–3866页。[24] I. A. S¸ucan、M. Moll 和 L. E. Kavraki，《开放运动规划库》，*IEEE*机器人与自动化杂志，第19卷，第4期，第72–82页，2012年12月。[25] B. Ichter、J. Harrison 和 M. Pavone，《学习机器人运动规划的采样分布》，载于*IEEE*机器人与自动化国际会议（*ICRA*）论文集，2018年，第7087–7094页。[26] F. Grothe、V. N. Hartmann、A. Orthey 和 M. Toussaint，《St-rrt*：穿越时空的渐近最优双向运动规划》，载于*IEEE*机器人与自动化国际会议（*ICRA*）论文集，2022年。