IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

PROTAMP-RRT: A Probabilistic Integrated Task and Motion Planner Based on RRT

Alessio Saccuti, *Graduate Student Member, IEEE*, Riccardo Monica, *Member, IEEE*, and Jacopo Aleotti, *Senior Member, IEEE*

***Abstract*—Solving complex robot manipulation tasks requires a Task and Motion Planner (TAMP) that searches for a sequence of symbolic actions, i.e. a task plan, and also computes collision-free motion paths. As the task planner and the motion planner are closely interconnected TAMP is considered a challenging problem. In this paper, a Probabilistic Integrated Task and Motion Planner (PROTAMP-RRT) is presented. The proposed method is based on a unified Rapidly-exploring Random Tree (RRT) that operates on both the geometric space and the symbolic space. The RRT is guided by the task plan and it is enhanced with a probabilistic model that estimates the probability of sampling a new robot configuration towards the next sub-goal of the task plan. When the RRT is extended, the probabilistic model is updated alongside. The probabilistic model is used to generate a new task plan if the feasibility of the previous one is unlikely. The performance of PROTAMP-RRT was assessed in simulated pick-and-place tasks, and it was compared against state-of-the-art approaches TM-RRT and Planet, showing better performance.**

***Index Terms*—Task and motion planning, manipulation planning.**

I. Introduction

PLANNING sequential robot manipulation tasks depends on both logical relations and geometric constraints. Task And Motion Planning (TAMP) approaches combine a task planner, which searches for a valid sequence of high-level symbolic actions (task plan), and a motion planner, which searches for a collision-free path to reach the goal [1]. The symbolic planner uses logical reasoning to determine a sequence of symbolic actions to reach the goal that are exploited to guide the motion planner. On the other hand, the motion planner may discover that some symbolic actions are not feasible due to robot kinematics constraints or collisions with obstacles and, therefore, the exploration of the configuration space can, in turn, affect the symbolic planner. In this paper, we propose a Probabilistic Integrated Task and Motion Planner (PROTAMP-RRT) based on a unified Rapidly-exploring Random Tree, i.e. an RRT that operates on both the geometric space and the symbolic space. The unified RRT algorithm, like a standard RRT motion planner [2], samples robot configurations, and it connects them in a tree data structure. However, as the distance between two configurations can not be easily defined if they refer to different high-level symbolic states, we propose to enhance the RRT with a probabilistic model which is exploited by the task planner to generate a sequence of symbolic actions to reach the goal of the task. Each symbolic action defines a sub-goal configuration of the robot. The RRT expansion is guided towards the sub-goal configuration of the first action in the task plan for which a path has not yet been found. The main contribution is the elaboration of the probabilistic model that evaluates the feasibility of the task plan, by estimating the probability that new collision-free configurations will be sampled. The probabilistic model is updated in the expansion phase of the RRT, and it is used to generate a new task plan if the feasibility of the previous one is unlikely. The probabilistic model takes into account probabilities conditioned on the pose of movable objects, so that the task planner can prevent unfavorable object configurations. PROTAMP-RRT defines symbolic actions using significant object poses, which are geometric poses of movable objects that are also represented as symbols. In order to limit computational load, PROTAMP-RRT starts with few significant object poses, but it can also add more of them if necessary. To do so, PROTAMP-RRT relies on sample generators which are able to extract new significant object poses, and the corresponding robot configurations to grasp and release objects in such poses. For example, if an object must be picked-up from a cupboard and an obstacle obstructs the extraction (Fig. 1) new significant object poses are generated to relocate the obstacle first. We evaluate PROTAMP-RRT on simulated pick-and-place tasks. Results indicate that PROTAMP-RRT outperforms two state-of-the art TAMP approaches: TM-RRT [3] and Planet [4]. In summary, the contributions of this work are: 1) a probabilistic feasibility model for TAMP, 2) a unified RRT approach that takes advantage of the probabilistic model and 3) the experimental evaluation against previous works.

II. Related Work

In most TAMP approaches, the task planner generates symbolic plans that are validated using a motion planner. Backtracking occurs when the motion planner fails, and new symbolic plans are generated. In the Task-Motion Kit [5] new symbolic

Manuscript received 28 April 2023; accepted 23 September 2023. Date of publication 25 October 2023; date of current version 9 November 2023. This letter was recommended for publication by Associate Editor Rahul Shome and Editor Hanna Kurniawati upon evaluation of the reviewers' comments. This work was supported by E80 Group S.p.A., Italy. *(Corresponding author: Alessio Saccuti.)*

The authors are with the Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy (e-mail: alessio.saccuti@unipr.it; riccardo.monica@unipr.it; jacopo.aleotti@unipr.it).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3327657, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3327657

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

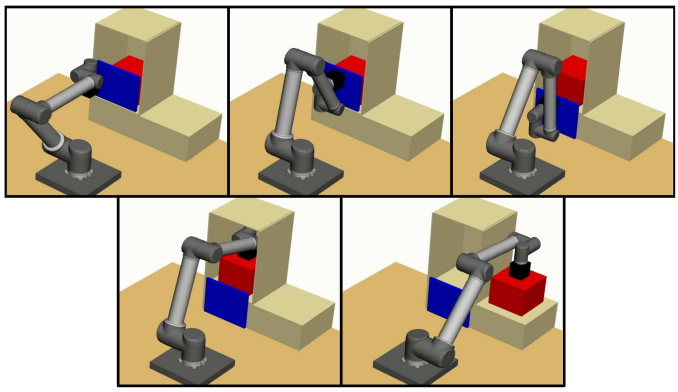

Fig. 1. Complex manipulation task where the red box must be extracted from a cupboard. Before the extraction, the blue panel must be relocated.

plans are iteratively deepened if the motion planner fails. In [6], sampled symbolic actions are validated by a RRT-Connect motion planner, until the task planner can build a complete symbolic plan. PDDLStream [7] by Garrett et al. introduces *streams*, i.e. external "black-box" algorithms which add new symbols to the symbolic space. A motion planner is a stream which generates trajectories for abstract actions. These approaches suffer from the fact that sampling-based motion planners, like RRT, may never terminate if the motion planning problem is not feasible. Therefore, these methods either use a simpler motion planner, or determine failures using a timeout, which depends on the complexity of the task.

Our approach of sampling new significant object poses is most similar to PDDLStream [7], where the pose generator is also a stream. Refinement functions in Task-Motion Kit [5] are a related concept, as they convert an abstract action into a geometric goal. Also semantic attachments [8] can produce new values for numeric fluents when actions are evaluated. The difference in our approach is that new significant object poses are sampled independently of existing actions.

As TAMP is computationally intensive, several methods have been proposed to guide task planning without using feedback from motion planning like our method. Heuristics have been proposed to estimate the action feasibility in the task planner, using a database [9], a SVM classifier [10] and reinforcement learning [11]. In order to help the task planner with geometric information, scene graphs [12] and neural networks [13] have also been proposed. Xu et al. [14] use deep learning to predict future long-term affordances in the task planner. A few works [15], [16] focused on transferring learned feasibility constraints from previous executions.

Similarly to our work, a few approached have used a unified RRT in combined symbolic and geometric space. The main challenge is guiding the RRT towards the goal, which requires long-horizon planning. In TM-RRT [3] by Caccavale and Finzi, RRT is guided by a distance function that accounts for the cardinality of the difference of two symbolic states. In Planet [4] a unified RRT is guided using a heuristic which takes into account failures due to collisions, but unlike in the proposed PROTAMP-RRT the heuristic does not consider the pose of the movable objects. Another work [17] extracts rich information from the geometric planner as it identifies "culprits", i.e., problematic sequences of actions. However, [17] operates on a discretized environment, and it defines culprits as logical statements. Similarly, [18] defines geometric constraints in a quantized representation of the parameters.

Some TAMP approaches focused on object re-arrangement in order to move or remove a single target object [19], [20], [21]. These methods can make additional assumptions, such as heuristics based on regions [20], and removal of objects to reduce constraints [21]. While our work aims at finding the first geometrically feasible solution, other works [22], [23], [24] focus on finding an optimal trajectory, e.g. in terms of robot execution time, by applying complex optimization approaches. The work in [25] deals with advanced sampling approaches for geometric locations where symbolic transitions may take place. In [26] general sampling-based methods are proposed for TAMP, but unlike our approach it focuses on sampling with dimensionality-reducing constraints.

III. Method

A planning problem is defined as the tuple $\langle \mathcal{E}, \mathcal{O}, \mathcal{S}, \mathcal{C}, \mathcal{M}, \mathcal{A}, \mathcal{R}, \mathcal{L}, \mathcal{G}, q_0, s_0 \rangle$, where $\mathcal{E}$ is the static environment, $\mathcal{O}$ is the set of manipulable objects, $\mathcal{S}$ is the symbolic state space of object predicates, $\mathcal{C} \subseteq \mathbb{R}^{\text{robot DoF}}$ is the robot configuration space, $\mathcal{M}$ are all possible *motions* connecting two robot configurations $q_1, q_2 \in \mathcal{C}$, $\mathcal{A}$ is the symbolic action space, $\mathcal{L}$ is a set of significant object poses, $\mathcal{R}$ is a set of regions where poses in $\mathcal{L}$ can be sampled, $\mathcal{G}$ is the task goal, $q_0 \in \mathcal{C}$ is the initial robot configuration, $s_0 \in \mathcal{S}$ is the initial symbolic state. A symbolic state $s \in \mathcal{S}$ may contain any predicate about objects. However, this work deals with predicates that specify the poses of the objects. The set $\mathcal{L}$ contains a finite number of significant poses of objects, i.e. geometric poses that are also represented as symbols in the task planner. In each symbolic state s objects are either assigned to a significant object pose in $\mathcal{L}$, or attached to the robot gripper. When an object is in a significant object pose the symbolic state may also change if the object is grasped or released by the robot. When a grasped object is moved by the robot, in-between poses are not encoded as significant object poses. Each object in $\mathcal{O}$ contains information about its geometry, the available grasping poses, and the regions $\mathcal{R}$ where it can be placed by the robot, either as intermediate locations or goal locations. Elements of $\mathcal{R}$ are continuous regions where placement of objects is allowed, e.g. flat surfaces. A sample generator that adds new significant object poses to $\mathcal{L}$ is associated to each region. The generators sample poses independently of actions or of existing object poses by extracting random poses in regions $\mathcal{R}$, with the requirement that at least one inverse kinematic solution exists to reach an object in that pose, but not that the pose is collision-free. Symbolic actions $a \in \mathcal{A}$ are operations that change the symbolic state, and in this work involve moving objects between significant poses. Task goal $\mathcal{G}$ is a set of conditions that must be satisfied by the final symbolic state.

The flowchart of the proposed planner is shown in Fig. 2. The main component is the RRT-based motion planner, which operates in an unified state space $\mathcal{X} = \mathcal{C} \times \mathcal{S}$ (Section III-A). The motion planner samples new unified states, checks them for collisions, and organizes them into a tree data structure $\mathcal{T}$.

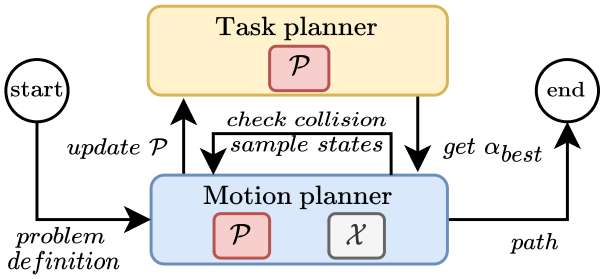

Fig. 2. PROTAMP-RRT flowchart.

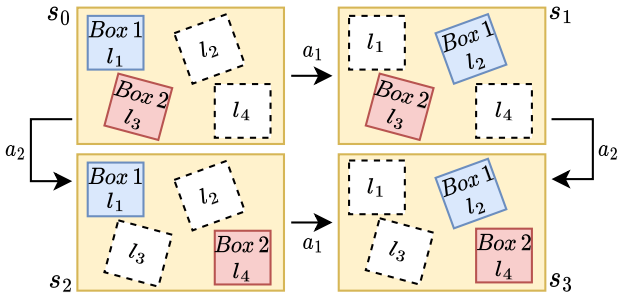

Fig. 3. Example symbolic states generated by applying actions a_1 and a_2 in different order to s_0. In s_0 (top left), $Box\ 1$ is in pose l_1, and $Box\ 2$ is in pose l_3. Two initially unoccupied poses are also defined: l_2 and l_4 (dashed).

A probabilistic model $\mathcal{P}$ (Section III-B) is updated whenever a new valid unified state is created or discarded due to collisions with the environment or other objects, in order to evaluate the probability that the motion plan will be able to progress towards the sub-goal for each action. If the probability of the current task plan becomes too low, the motion planner triggers the task planner (Section III-C) to obtain a new task plan, i.e. the sequence of actions α_{best} that reaches the goal $\mathcal{G}$ with the highest probability. The motion planner descends $\mathcal{T}$ until it finds the first unreached sub-goal of α_{best}, and then it extends $\mathcal{T}$ towards this sub-goal (Section III-D).

A. Unified State Space

Each symbolic state $s{\in}\mathcal{S}$ is a set of predicates that describe the environment. In this work predicates describe mainly the poses of movable objects. A symbolic state s contains a set of pairs c_i in the form $\langle o_i, l_{j(i)}\rangle$ that associates each manipulable object in $o_i \in \mathcal{O}$ to a (significant) object pose in $l_j{\in}\mathcal{L}$ (except for the grasped object, if any). The task planner operates on $\mathcal{S}$, where symbolic states are connected by abstract actions in $\mathcal{A}$. Each action $a{\in}\mathcal{A}$ has preconditions that must be verified before the action can be applied to the symbolic state, and post-conditions that describe the changes applied to the symbolic state. Postconditions include as a sub-goal the final configuration $q_{end}(a)$ of the robot after the action. Given a sequence of actions α and the initial symbolic state s_0, postconditions can be applied consecutively to determine a sequence of symbolic states $\{s_0, \ldots, s_i, \ldots, s_{|\alpha|}\}$, where s_i is the symbolic state after action a_i. Some actions in $\mathcal{A}$ are based on action templates which depend on locations and/or objects, such as $Move(o, l_\mathrm{s}, l_\mathrm{d})$ that moves object $o \in \mathcal{O}$ from l_s to l_d, where $l_\mathrm{s}, l_\mathrm{d} \in \mathcal{L}$ are significant object poses. When new significant object poses are sampled by the sample generators, new actions are added to $\mathcal{A}$ using the new poses.

Four examples of symbolic states are shown in Fig. 3: Initial symbolic state s_0 and states s_1, s_2, and s_3, where there is a different assignment of each box to the significant object poses $l_1, l_2, l_3, l_4 \in \mathcal{L}$. Two actions are defined: action $a_1 \in \mathcal{A}$ moves $Box\ 1$ from significant object pose l_1 to l_2, and action a_2 moves $Box\ 2$ from significant object pose l_3 to l_4. Symbolic state s_1 is generated by applying a_1 to the initial state s_0, and symbolic state s_2 by applying a_2. Both actions in any order lead to the same final symbolic state s_3.

The RRT motion planner operates on a unified state space $\mathcal{X} = \mathcal{C} \times \mathcal{S}$. Each unified state $x \in \mathcal{X}$ consists of a single robot configuration $q \in \mathcal{C}$ and a symbolic state $s \in \mathcal{S}$. The RRT based algorithm samples unified states $x \in \mathcal{X}$ and it organizes them in a tree data structure $\mathcal{T}$, with root at the initial unified state $x_0{=}(q_0, s_0)$ and whose edges are motions $m \in \mathcal{M}$. The distance between two unified states, in the same symbolic state, is computed as the Euclidean distance in the robot joint space. Motions may also include changes in the symbolic state. Therefore, given a motion, the sequence of symbolic actions performed up to that motion can be retrieved by traversing the tree upwards to the root. In practice, as the planner needs to efficiently query the sequence of symbolic actions which lead to each motion, each motion holds a reference to the sequence of actions.

B. Probabilistic Model

The purpose of the probabilistic model $\mathcal{P}$ is to evaluate the probability that the RRT planner will be able to progress towards the solution of an action $a \in \mathcal{A}$, given the knowledge about past performance of the motion planner on the same action. In particular, the probabilistic model is used to estimate the probability $\mathcal{P}(a|s)$ of action a in symbolic state s as the probability of successfully sampling a new robot configuration towards the action sub-goal $q_{end}(a)$. The new configuration is successfully sampled if it does not collide against the static environment or other movable objects. Therefore, under the assumption of independence of the c_i effects, $\mathcal{P}(a|s)$ may be decomposed as:

$$\mathcal{P}(a|s) = \mathcal{P}_e(a)\,\mathcal{P}_{obj}(a|s) = \mathcal{P}_e(a) \prod_{c_i \in s} \mathcal{P}_c\,(a|c_i) \qquad (1)$$

where $\mathcal{P}_e(a)$ is the probability that the new configuration does not collide with the static environment, and $\mathcal{P}_{obj}(a|s)$ is the probability that it does not collide with movable objects. In turn, $\mathcal{P}_{obj}(a|s)$ is the product of all $\mathcal{P}_c(a|c_i)$ where $c_i{=}\langle o_i, l_{j(i)}\rangle$ is the probability that the new configuration does not collide with object o_i if the latter is in pose $l_{j(i)}$. As $\mathcal{P}_c$ in (1) depends on the action and only on a single pair c_i, it is used to compute $\mathcal{P}(a|s)$ for all symbolic states s where c_i is true. In summary, the probabilistic model $\mathcal{P}$ has three elements for each action a: an *environmental probability* $\mathcal{P}_e(a)$, a set of *object conditional probabilities* $\mathcal{P}_C(a){=}\{\mathcal{P}_c(a|c_i)\}$, and a set of *solved symbolic states* $\mathcal{P}_S(a)$.

1) Environmental Probability: the environmental probability $\mathcal{P}_e(a)$ is the probability of not colliding with the static environment when sampling a robot configuration towards the action sub-goal $q_{end}(a)$. The probability is estimated as the fraction of configurations sampled towards the action sub-goal which did not collide with the environment:

$$\mathcal{P}_e(a) = \frac{N_e(a)}{D_e(a)} \tag{2}$$

where N_e and D_e are two positive integers. Initially, $N_e(a){=}N_0^e$, $D_e(a){=}D_0^e$, where N_0^e and D_0^e are positive integer constants and $N_0^e{<}D_0^e$. Whenever the RRT discards a sampled configuration due to a collision with the static environment $\mathcal{E}$, $\mathcal{P}_e(a)$ is reduced by incrementing D_e. Whenever the RRT successfully samples a configuration towards the action sub-goal $q_{end}(a)$, $\mathcal{P}_e(a)$ is incremented by incrementing both $N_e(a)$ and $D_e(a)$. After a successful sampling, if $\mathcal{P}_e(a){<}\frac{N_{\min}}{D_{\min}}$ we reset $N_e(a){=}N_{\min}$ and $D_e(a){=}D_{\min}$, where $N_{\min}$ and $D_{\min}$ are constants ($N_{\min}{<}D_{\min}$). This rule restores the probability to a higher value when the RRT samples a robot configuration towards an action sub-goal after many failures, e.g. when a narrow passage is found after many collisions.

2) Object Conditional Probabilities: the set of object conditional probabilities $P_C(a)$ for action a defines a probability $\mathcal{P}_c(a|c_i)$ for each object/pose pair $c_i = \langle o_i, l_{j(i)} \rangle$ in a set $C(a)$ of object/pose pairs. A $\mathcal{P}_c(a|c_i)$ represents the probability that a robot configuration does not collide with object o_i in significant pose l_j. Set $C(a)$ is initially empty, and it is filled with object/pose pairs that negatively affect action a as they cause collisions. Similarly to the environmental probability, the object conditional probability is estimated as the fraction of configurations extracted towards the action sub-goal that did not collide with the object o_i when it was at location $l_{j(i)}$:

$$\forall c_i \in C(a) \qquad \mathcal{P}_c\left(a|c_i\right) = \frac{N_c\left(a|c_i\right)}{D_c\left(a|c_i\right)} \tag{3}$$

where N_c and D_c are two positive integers. Whenever the RRT fails to sample a robot configuration due to a collision, all colliding objects and their poses are collected in a set of pairs $C_{new} = \{c_i\}$. For each $c_i \in C_{new}$, if c_i is not already in $C(a)$, it is added and $\mathcal{P}_c(a|c_i)$ is initialized with $N_c(a|c_i) = N_0^c$, $D_c(a|c_i) = D_0^c$. Otherwise, if c_i is in $C(a)$, $\mathcal{P}_c(a|c_i)$ is decreased by incrementing $D_c(a|c_i)$. Conversely, if the RRT succeeds in sampling a configuration towards $q_{end}(a)$, probability $\mathcal{P}_c(a|c_i)$ is increased by incrementing $N_c(a|c_i)$ and $D_c(a|c_i)$ for all $c_i \in C(a)$. Similarly to $\mathcal{P}_e(a)$, after a successful sampling, if $\mathcal{P}_c(a|c_i) < \frac{N_{\min}}{D_{\min}}$, then we reset $N_c(a|c_i) = N_{\min}$ and $D_c(a|c_i) = D_{\min}$.

3) Solved Symbolic States: a set of symbolic states $\mathcal{P}_S(a)$ where the action sub-goal $q_{end}(a)$ was reached for action a. That is, when the RRT reaches $q_{end}(a)$ in a symbolic state s, then s is added to $\mathcal{P}_S(a)$.

When the probabilistic model is queried to obtain probability $\mathcal{P}(a|s)$, first it checks if s is among solved symbolic states $\mathcal{P}_S(a)$, as in this case the action sub-goal has already been reached and probability is 1. Else, probability $\mathcal{P}(a|s)$ is computed according to (1), (2) and (3).

Algorithm 1: PROTAMP-RRT Planning Algorithm.

Input: $\mathcal{E}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{L}, \mathcal{G}, q_0, s_0$
Output: $\alpha_{best}, \mathcal{T}$
$\mathcal{P} \leftarrow \varnothing, \mathcal{T} \leftarrow \{(s_0, q_0)\}, P_{best} = 1$
while true **do**
▷ Task planning
  if $\alpha_{best} = \varnothing$ or $\mathcal{P}(\alpha_{best}) < P_{best}\, P_{mult}$ **then**
    $\alpha_{best} \leftarrow TaskPlanner(\mathcal{O}, \mathcal{A}, \mathcal{L}, \mathcal{G}, s_0, \mathcal{P})$
    $r \leftarrow ReachableActions(\mathcal{T}, \alpha_{best})$
    while $\mathcal{P}_{obj}(\alpha_{best}) \leq P_{low}^{|\alpha_{best}|-(r-1)}$ **do**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{SampleNewObjectPose(\mathcal{R})\}$
      $\alpha_{best} \leftarrow TaskPlanner(\mathcal{O}, \mathcal{A}, \mathcal{L}, \mathcal{G}, s_0, \mathcal{P})$
      $r \leftarrow ReachableActions(\mathcal{T}, \alpha_{best})$
    $P_{best} \leftarrow \mathcal{P}(\alpha_{best})$
  $q_{goal} \leftarrow q_{end}(a_r)$ ▷ Motion planning
  $q_{new} \leftarrow SampleState(\mathcal{T}, q_{goal})$
  $C_{new} \leftarrow CheckCollisions(q_{new}, s_{r-1})$
  $\mathcal{P} \leftarrow UpdateProb(\mathcal{P}, \mathcal{E}, \mathcal{T}, a_r, q_{new}, q_{goal}, C_{new}, s_{r-1})$
  if $C_{new} = \varnothing$ **then**
    $\mathcal{T} \leftarrow Extend(\mathcal{T}, (s_{r-1}, q_{new}))$
    if $q_{new} = q_{goal}$ and $r = |\alpha_{best}|$ **then**
      return $\alpha_{best}, \mathcal{T}$
    if $q_{new} = q_{goal}$ **then**
      $SetSolved(\mathcal{P}, a_r, s_{r-1})$
      $r \leftarrow r + 1$

C. Task Planner

The task planner searches for a task plan, i.e. a sequence of symbolic actions $\alpha_{best} = \{a_1, a_2, \ldots, a_{|\alpha_{best}|}\}$ that satisfies the task goal $\mathcal{G}$ while maximizing $\mathcal{P}(\alpha)$, i.e., the joint probability of performing the entire sequence of actions α:

$$\mathcal{P}\left(\alpha\right) = \prod_{i=1,\ldots,|\alpha|} \mathcal{P}\left(a_i|s_{i-1}\right) \tag{4}$$

where s_{i-1} is the symbolic state obtained after applying all actions of the task plan up to a_{i-1}. In this work, a tree search approach was used based on $A^\star$, which prioritizes the expansion of the node with the highest value of the heuristic:

$$F\left(\alpha'\right) = \mathcal{P}\left(\alpha'\right)\, H\left(s_{|\alpha'|}\right) \tag{5}$$

where α' is the partial plan up to that node, and $H(s_{|\alpha'|})$ estimates the probability to reach the task goal given the final symbolic state $s_{|\alpha'|}$ of α'. For each condition in $\mathcal{G}$ (e.g., an object must be in a given location) which is violated by state $s_{|\alpha'|}$, at least one additional action must be taken by the robot. Hence, given a constant P_H defining the a-priori probability of an action, heuristic $H(s_{|\alpha'|})$ is:

$$H(s_{|\alpha'|}) = (P_H)^{U(s_{|\alpha'|})} \tag{6}$$

where $U(s_{|\alpha'|})$ is the number of unsatisfied task goal conditions. A bounded approach was used to speed up the search after the first execution of the task planner, given the previous solution α_{prev}. Nodes where $\mathcal{P}(\alpha') < \mathcal{P}(\alpha_{prev})$ are pruned, as they cannot lead to an improvement over α_{prev}.

Algorithm 2: Probability Update Procedure $UpdateProb$.

Input: $\mathcal{P}, \mathcal{E}, \mathcal{T}, a, q_{new}, q_{goal}, C_{new}, s$
Output: $\mathcal{P}$
1: **if** $q_{new} = q_{goal}$ and $C_{new} \neq \varnothing$ **then** ▷ Goal unreachable
2: $\quad \forall c_i \in C_{new}, \quad \mathcal{P}_c(a|c_i) \leftarrow 0$
3: **else**
4: $\quad$ **if** $C_{new} \neq \varnothing$ **then** ▷ Collision
5: $\quad\quad$ **if** $\mathcal{E} \in C_{new}$ **then** $Decrease(\mathcal{P}_e(a))$
6: $\quad\quad \forall c_i \in C_{new}, \quad Decrease(\mathcal{P}_c(a|c_i))$
7: $\quad$ **if** $C_{new} = \varnothing$ and ▷ Success
8: $\quad\quad \forall q|(q,s) \in \mathcal{T}, \quad |q_{new} - q_{goal}| < |q - q_{goal}|$ **then**
9: $\quad\quad Increase(\mathcal{P}_e(a))$
10: $\quad\quad \forall c_i \in s \cap C(a), \quad Increase(\mathcal{P}_c(a|c_i))$

D. Integrated Task and Motion Planner

PROTAMP-RRT searches for a solution with a single execution of the unified RRT planner. The unified RRT motion planner iteratively extends $\mathcal{T}$ until a path to a symbolic state that satisfies the task goal $\mathcal{G}$ is found. The extension is guided by the most promising sequence of symbolic actions α_{best}.

The planning algorithm (Algorithm 1) has a main loop (lines 2–22) that consists of two parts, task planning and motion planning. The task planning part (lines 4–11) invokes the task planner to obtain α_{best}, while the motion planning part (lines 12–19) extends the RRT $\mathcal{T}$ guided by the symbolic plan.

Task planning (line 5) is executed only if there is no symbolic plan α_{best}, or if its probability $\mathcal{P}(\alpha_{best})$ (4) falls below a threshold $P_{best}\, P_{mult}$, where $P_{mult} \ll 1$ is a constant and P_{best} (line 11) is the probability of the plan when it was first generated. Once $\alpha_{best} = \{a_1, \ldots, a_{|\alpha_{best}|}\}$ is available, the symbolic states traversed by α_{best}, $\{s_0, \ldots, s_{|\alpha_{best}|}\}$, are known. The planner uses $ReachableActions$ (line 6) to descend the motion tree $\mathcal{T}$ and to find the index r of the first action a_r whose sub-goal has not been reached yet by the RRT. Symbolic state s_{r-1} is obtained by applying all actions of α_{best} up to a_{r-1}. If the task planner cannot find a plan with probability greater than a threshold (line 7), a new significant object pose is sampled into $\mathcal{L}$ from each region in $\mathcal{R}$ (line 8), thus adding new actions until a new plan becomes feasible. As new actions are added with an environmental probability $\mathcal{P}_e < 1$, old actions with high probability may still be preferred by the task planner. Sampling new object poses never influences environmental probability, hence the test at line 7 uses only object conditional probability of the plan, defined as:

$$\mathcal{P}_{obj}\left(\alpha_{best}\right) = \prod_{i=1}^{r} \mathcal{P}_{obj}\left(a_i|s_{i-1}\right) \tag{7}$$

where $\mathcal{P}_{obj}(a|s)$ is defined in (1). The dynamic threshold (line 7) is a constant P_{low} raised to the number of non-reached sub-goals of α_{best}. At each iteration, the motion planner first sets the current sub-goal q_{goal} to the sub-goal $q_{end}(a_r)$ of action a_r (line 12). Then, at line 13 a new configuration q_{new} is sampled by $SampleState$, randomly or towards the action sub-goal with probability P_{goal}. When $SampleState$ is called and q_{goal} is different from the previous iteration of the main loop, $SampleState$ always extracts q_{goal}, to check if the action is trivially feasible or infeasible. The collision check is performed at line 14 and a set of pairs $C_{new} = \{\langle o_i, l_{j(i)}\rangle\}$ of colliding objects and their locations are found. This set is used to update the probabilistic model $\mathcal{P}$ ($UpdateProb$, line 15), which in turn changes the symbolic plan probability $\mathcal{P}(\alpha_{best})$ for the next iteration. If no collisions are found (line 16) $\mathcal{T}$ is extended with q_{new} (line 17). If $\mathcal{T}$ has been extended to q_{goal} and r was the index of the last action of α_{best}, the algorithm terminates because the task goal $\mathcal{G}$ is reached (lines 18–19). Otherwise, if $\mathcal{T}$ has been extended to q_{goal}, but other actions must be performed, the symbolic state s_{r-1} is added to the set of solved symbolic states $\mathcal{P}_S(a_r)$, and $\mathcal{P}_e(a_r)$ is set to 1 in the probabilistic model ($SetSolved$ at line 21). Also, as q_{goal} has been reached, the next action of α_{best} is reachable, and therefore r is incremented (line 22). Then, the algorithm continues with the next iteration of the main loop.

In $UpdateProb$ (Algorithm 2) if the sampled state q_{new} is the action sub-goal and it is in collision (line 1), then the action is set as infeasible whenever any colliding objects/pose pairs are present in C_{new} (line 2). If sampled state q_{new} is not the action sub-goal and it is in collision (lines 4–6), then the obstacles are hindering the action and probabilities $\mathcal{P}_e(a)$ and $\mathcal{P}_c(a|c_i)$ are decreased as explained in Sections III-B1 and III-B2. Probabilities are increased (lines 7–10) if q_{new} is not in collision, and if it is the nearest configuration to q_{goal} among the configurations in $\mathcal{T}$ with symbolic state s.

The $UpdateProb$ procedure provides a negative feedback if the RRT fails to expand towards the current action sub-goal. In particular, in Algorithm 2 (line 8) the probability is increased only if the new configuration is closer to the action sub-goal than each previous configuration. The probability of finding configurations which are closer to the action sub-goal decreases exponentially with the number of configurations in $\mathcal{T}$. While expansion towards the same action sub-goal is attempted and fails, the probability $\mathcal{P}(\alpha_{best})$ is reduced and it will eventually be lower than P_{mult} multiplied by its initial value. When this occurs, the task planner is called again, to find a (potentially) different plan with higher probability.

IV. Experiments

A. Experimental Setup

The PROTAMP-RRT approach was evaluated in simulated manipulation tasks in which a robot moves boxes. The robot is a fixed Universal Robot UR10 manipulator (with a reach of 1300 mm). A parallelepiped of size $0.1 \times 0.1 \times 0.11\,\mathrm{m}$ was used to simulate an OnRobot VGC10 vacuum gripper. We assumed that gripper is able to perform grasping by contact with any flat side of the objects, disregarding dynamic effects. The simulated experimental environment was implemented on top of ROS (Robot Operating System) and visualized in RViz. The software was implemented in C++, using the OMPL planning library [27]. The MoveIt planning framework and the Flexible Collision Library [28] were used to instantiate the geometry and check for collisions. The IKFast OpenRAVE

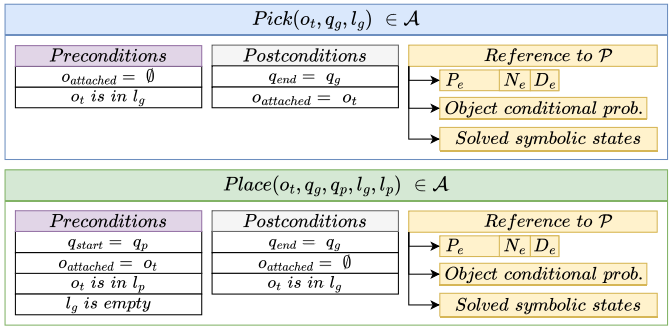

Fig. 4. Definition of *Pick* and *Place* symbolic actions.

module was used to create an analytical inverse kinematics solver for the UR10. The experimental evaluation ran on an Intel i9 10900 processor @ 2.80 GHz (32 GB of RAM, Ubuntu 20.04 LTS). PROTAMP-RRT was configured with $N_0^e = 45$, $D_0^e = 50$, $N_0^c = 49$, $D_0^c = 50$, $N_{min} = 30$, $D_{min} = 50$, $P_H = 0.815$, $P_{mult} = 0.1$, $P_{low} = 0.8$, $P_{goal} = 0.3$.

B. Manipulation Tasks

We tested PROTAMP-RRT on simulated pick-and-place tasks. Two action templates were considered, i.e. *pick* and *place* (Fig. 4). For each action, the probabilistic model $\mathcal{P}$ defines the environmental probability, the object conditional probabilities, and the set of solved symbolic states. A *pick action* represents the action of approaching an object while the gripper is empty, and then attaching the object to the gripper. This action is parameterized by the target object o_t, the significant object pose l_g of o_t, and the robot configuration q_g to grasp the object in l_g. For each object o_t and pose l_g, multiple pick actions are generated with different configurations q_g, as the robot may grasp the object from multiple sides and each side may be reached with a different inverse kinematic solution. The preconditions imply that no object is attached to the gripper ($o_{attached} = \emptyset$) and o_t is in l_g. As result of the action, the final robot configuration q_{end} is q_g and o_t is attached to the end effector. A *place action* represents the action of moving a grasped object to a new location, and then releasing the object. This action is parameterized by the target object o_t, the initial significant object pose l_p, the final significant object pose l_g, the initial robot configuration q_p, and the final robot configuration, q_g. As for the pick actions, for each o_t, l_p and l_g, multiple place actions are generated with different configurations q_p and q_g, taking into account multiple inverse kinematic solutions. Preconditions are that the initial robot configuration is q_p, o_t is grasped and it is in significant object pose l_p, while l_g pose must not be occupied by other objects. In the postconditions, the final robot configuration q_{end} is q_g, o_t is in l_g and no object is grasped.

In total, 6 manipulation tasks were simulated (Fig. 5, also shown in the attached video). In the *Transfer 1*, *Transfer 2* and *Transfer 3* tasks, red objects O_1, O_2 and O_3 must be relocated into significant object poses sampled from the *Place table* planar region. In *Transfer 1*, no obstacles are present between initial and goal locations, while in *Transfer 2* and *Transfer 3* walls have been added around the objects. The robot is allowed to grasp objects

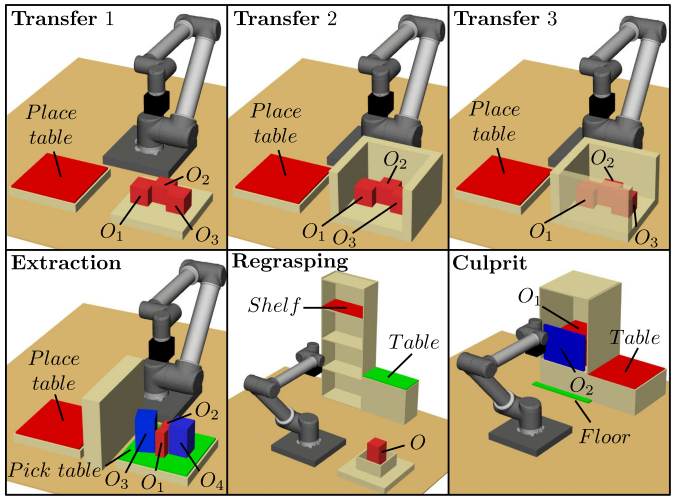

Fig. 5. Simulated manipulation tasks, from the top left: *Transfer 1*, *Transfer 2*, *Transfer 3*, *Extraction*, *Regrasping* and *Culprit*.

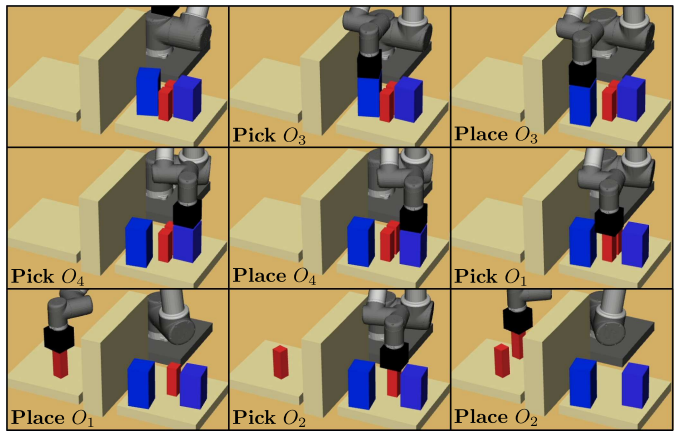

Fig. 6. Example of *Extraction* task execution.

only from above and significant object poses can be sampled in the *Place table* region only. In the *Extraction* task (Fig. 6), red objects O_1 and O_2, must be relocated into the *Place table* region. Initially, grasping is infeasible due to the obstructing objects (blue boxes), O_2 and O_3, which have to be relocated. The robot can grasp each movable object from above or from the front side. Significant object poses for the red objects can be sampled in the *Place table* region, while for the blue objects they can be sampled in the *Pick table* region. In the *Regrasping* task the red box O must be relocated on top of the *Shelf* region. This task requires regrasping as initially the object can be picked only from above, and it can be placed in the deposit region *Shelf* only if it is grasped from the side, due to obstacles. Movable objects can be grasped from above or from the front side. Significant object poses can be sampled in the *Shelf* region, and in the *Table* region where re-grasping may occur.

Finally, in the *Culprit* task red box O_1 must be extracted from the cupboard and relocated in the *Table* region. To do this, panel O_2 must be removed, as the opening is slightly too small otherwise. This task is challenging as the planner can be stuck in a strong local maximum: the robot is able to pick and

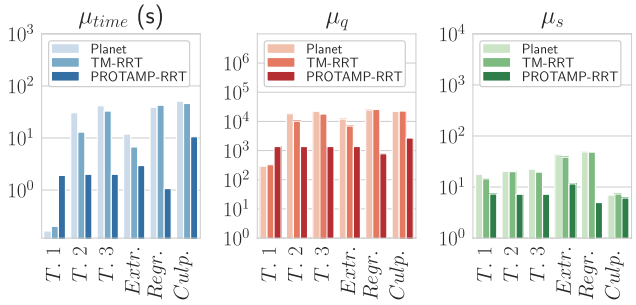

Fig. 7. Avg. planning time (μ_{time}), avg. number of robot configurations (μ_q), avg. number (μ_s) of symbolic states (for Planet, TM-RRT, PROTAMP-RRT).

move O_1 within the cupboard, and the planner can therefore successfully sample many robot configurations without moving O_2, however, none of these samples lead to a solution of the task. Significant object poses for the red object can be sampled in the *Table* region, and for O_2 they can be sampled in the *Floor* region. An example is shown in Fig. 1.

C. Experiments in Finite Symbolic Space

Our method was compared against two other unified RRT-based approaches, TM-RRT [3] and Planet [4], on the six manipulation tasks: *Transfer 1* (*T* 1), *Transfer 2* (*T* 2), *Transfer 3* (*T* 3), *Extraction* (*Extr.*), *Regrasping* (*Regr.*) and *Culprit* (*Culp.*). As TM-RRT can operate only on a finite symbolic space, we pre-defined a minimal finite set of significant object poses sufficient to solve the planning problem. Hence, sampling of new significant object poses in PROTAMP-RRT was disabled to ensure fair comparison. For the same reason, the goal of each task was not defined in terms of regions, but as a specific goal pose for each object. Planet was also modified so that actions did not sample their sub-goal randomly, but selected it among the pre-sampled poses. For each task and each method, planning was executed 50 times.

The average planning time, the average number of robot configurations and the average number of symbolic states in $\mathcal{T}$ are in Fig 7. Planet and TM-RRT sampled more unified states in $\mathcal{X}$, as evidenced by the higher values of μ_q and μ_s. Instead, thanks to the probabilistic model $\mathcal{P}$, PROTAMP-RRT focused on the symbolic actions with highest probability, and therefore resulted in a lower planning time μ_{time}. The only exception is *Transfer 1* task, where no obstacles were present and motion planning was trivial. Although in *Transfer 1* PROTAMP-RRT still explored fewer symbolic states (lower value of μ_s), both Planet and TM-RRT resulted in a lower planning time μ_{time}. The main reason is that these algorithms use heuristics to guide the RRT instead of a long-horizon task planner.

The success rate as a function of planning time is shown in Fig. 8. Planet and TM-RRT success rate is generally lower than PROTAMP-RRT. At the maximum tested time limit of 600 seconds, Planet reached 100% success rate only in *Transfer 1* and *Extraction*, while TM-RRT only in *Transfer 1*, *Transfer 2*, *Transfer 3* and *Extraction*. In particular, Planet and TM-RRT struggled the most in the *Regrasping* and *Culprit* tasks. Instead, our method reached 100% success rate in all tasks except the

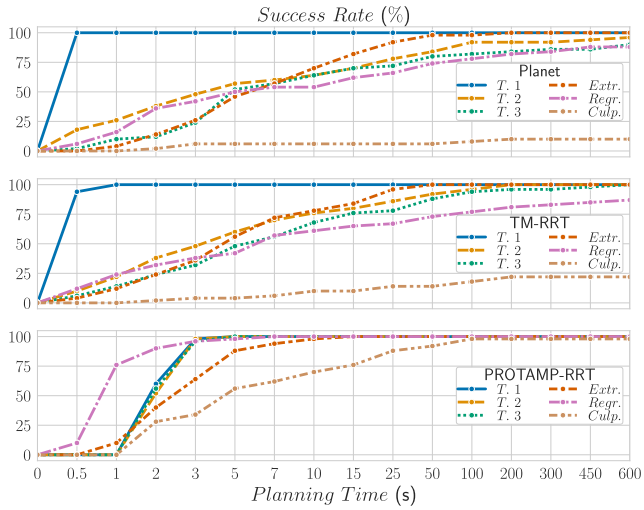

Fig. 8. Planet, TM-RRT and PROTAMP-RRT average success rate with respect to planning time limit, with a finite symbolic space.

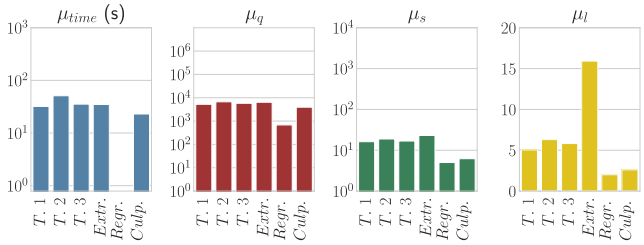

Fig. 9. PROTAMP-RRT in unlimited symbolic space. Left to right: Avg. planning time (μ_{time}), avg. number of robot configurations (μ_q), avg. number of symbolic states (μ_s), avg. number of sampled significant poses (μ_l).

Culprit task, where success rate was 98%, compared to the 22% of TM-RRT and 10% of Planet. The proposed probabilistic model is very effective in the solution of the *Culprit* task, as it is able to discover that, even if the red box can be picked and moved within the cupboard, it can not be extracted until the blue panel is moved.

D. Experiments in Unlimited Symbolic Space

In this section, the full PROTAMP-RRT algorithm is evaluated in an unlimited symbolic space, without a pre-defined set of significant object poses, so that it sampled new significant object poses to reach the goal. Results are shown in Fig. 9. Compared to PROTAMP-RRT in Fig. 7, where the significant object poses were pre-defined, the higher complexity of the task resulted in an increased planning time μ_{time}, number of robot configurations μ_q and number of symbolic states μ_s, by about an order of magnitude. The only exception is the *Regrasping* task, where the results are similar.

Fig. 9 also reports the average number of sampled significant object poses (μ_l). As new significant poses are sampled only when task plan probability is very low, PROTAMP-RRT limited the expansion of the symbolic space. The task where more significant object poses were sampled is the *Extraction* task, where four movable objects were present. However, not all resulting symbolic states were explored and added to $\mathcal{T}$ as

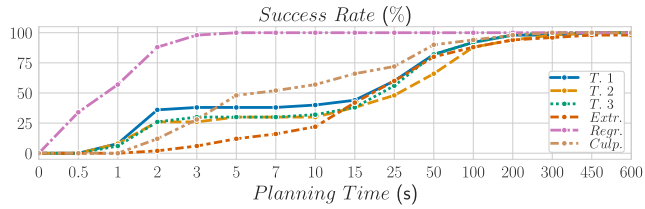

Fig. 10. PROTAMP-RRT average success rate in unlimited symbolic space.

evidenced by μ_s, which is comparable to μ_s for the *Transfer* tasks. This is likely an effect of the probabilistic model, that determined which symbolic states should be avoided. Fig. 10 shows PROTAMP-RRT success rate as a function of planning time limit. Even if results are worse than in Fig. 8, PROTAMP-RRT successfully solved most of the trials after about 400 seconds. Results confirm that the *Extraction* task seems to be the most challenging.

Further experiments and data are available at https://rimlab.ce.unipr.it/Saccuti.html, in the Material section.

V. Conclusion

An integrated TAMP approach based on a unified Rapidly-exploring Random Tree was proposed, that operates on both the geometric space and the symbolic space. The method introduces a probabilistic model that estimates the feasibility of symbolic actions by evaluating the probability that new collision-free configurations will be sampled. The PROTAMP-RRT method was evaluated in several manipulation tasks showing better performance than TM-RRT and Planet in terms of planning time and success rate. Moreover, the ability of PROTAMP-RRT to expand the symbolic space was evaluated indicating that it is able to solve planning problems by sampling a few new significant object poses. Future work will involve the investigation of a more advanced symbolic planner to evaluate logical statements other than object/pose pairs.

References

[1] C. R. Garrett et al., "Integrated task and motion planning," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, no. 1, pp. 265–293, 2021.
[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Res. Tech. Rep. 9811, 1998.
[3] R. Caccavale and A. Finzi, "A rapidly-exploring random trees approach to combined task and motion planning," *Robot. Auton. Syst.*, vol. 157, 2022, Art. no. 104238.
[4] W. Thomason and R. A. Knepper, "A unified sampling-based approach to integrated task and motion planning," in *Proc. Int. Symp. Robot. Res.*, 2019, pp. 773–788.
[5] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "The task-motion kit: An open source, general-purpose task and motion-planning framework," *IEEE Robot. Automat. Mag.*, vol. 25, no. 3, pp. 61–70, Sep. 2018.
[6] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *Int. J. Robot. Res.*, vol. 37, no. 1, pp. 104–136, 2018.
[7] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2018, pp. 440–448.
[8] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *Proc. 19th Int. Conf. Automated Plan. Scheduling*, 2009, pp. 114–121.
[9] P. M. U. Eljuri, G. A. G. Ricardez, N. Koganti, J. Takamatsu, and T. Ogasawara, "Combining a Monte Carlo tree search and a feasibility database to plan and execute rearranging tasks," *IEEE Access*, vol. 9, pp. 21721–21734, 2021.
[10] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1255–1262, Apr. 2019.
[11] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Proc. Conf. Robot Learn.*, 2020, pp. 955–968.
[12] Z. Jiao, Y. Niu, Z. Zhang, S.-C. Zhu, Y. Zhu, and H. Liu, "Sequential manipulation planning on scene graph," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 8203–8210.
[13] M. Khodeir, B. Agro, and F. Shkurti, "Learning to search in task and motion planning with streams," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 1983–1990, Apr. 2023.
[14] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 6206–6213.
[15] T. Ren, A. I. Cowen-Rivers, H. Bou-Ammar, and J. Peters, "Learning geometric constraints in task and motion planning," 2022, *arXiv:2201.09612*.
[16] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *Int J. Robot. Res.*, vol. 38, no. 7, pp. 793–812, 2019.
[17] F. Lagriffoul and B. Andres, "Combining task and motion planning: A culprit detection problem," *Int. J. Robot. Res.*, vol. 35, no. 8, pp. 890–927, 2016.
[18] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proc. IEEE/RSJ Intl Conf. Intell. Robots Syst.*, 2014, pp. 3684–3691.
[19] C. Nam, S. H. Cheong, J. Lee, D. H. Kim, and C. Kim, "Fast and resilient manipulation planning for object retrieval in cluttered and confined environments," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1539–1552, Oct. 2021.
[20] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 6621–6627.
[21] A. Krontiris and K. Bekris, "Trade-off in the computation of minimum constraint removal paths for manipulation planning," *Adv. Robot.*, vol. 31, pp. 1–12, 2017.
[22] N. Castaman, E. Tosello, and E. Pagello, "Conditional task and motion planning through an effort-based approach," in *Proc. IEEE Int. Conf. Simul., Model., Program. Auton. Robots*, 2018, pp. 49–54.
[23] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4044–4051.
[24] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and motion informed trees (TMIT*): Almost-surely asymptotically optimal integrated task and motion planning," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11370–11377, Oct. 2022.
[25] J. Ortiz-Haro, V. N. Hartmann, O. S. Oguz, and M. Toussaint, "Learning efficient constraint graph sampling for robotic sequential manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4606–4612.
[26] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *Int. J. Robot. Res.*, vol. 37, no. 13/14, pp. 1796–1825, 2018.
[27] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012. [Online]. Available: https://ompl.kavrakilab.org
[28] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 3859–3866.