Anticipatory Task and Motion Planning: Improved Rearrangement in Persistent Continuous-Space Environments

Roshan Dhakal, Duc M. Nguyen, Tom Silver, Xuesu Xiao, *Member, IEEE*, and Gregory J. Stein, *Member, IEEE*

***Abstract*—We consider a sequential task and motion planning (TAMP) setting in which a robot is assigned continuous-space rearrangement-style tasks one-at-a-time in an environment that persists between each. Lacking advance knowledge of future tasks, existing (myopic) planning strategies unwittingly introduce side effects that impede completion of subsequent tasks: e.g., by blocking future access or manipulation. We present *anticipatory task and motion planning*, in which estimates of expected future cost from a learned model inform selection of plans generated by a model-based TAMP planner so as to avoid such side effects, choosing configurations of the environment that both complete the task and reduce overall cost. Simulated many-task deployments in navigation-among-movable-obstacles and cabinet-loading domains yield improvements of 32.7% and 16.7% average per-task cost respectively. When given time in advance to *prepare* the environment, our learning-augmented planning approach yields improvements of 83.1% and 22.3%. Finally, we also demonstrate anticipatory TAMP on a real-world Fetch mobile manipulator.**

***Index Terms*—Integrated planning and learning, task and motion planning.**

I. Introduction

We consider a sequential task and motion planning (TAMP) setting, in which a long-lived robot is assigned rearrangement-style tasks one-at-a-time from a sequence. The environment persists between tasks, so that the terminal state after completing one task serves as the starting state for the next. Lacking advance knowledge of what tasks the robot will later be assigned, existing TAMP planners [1], [2], [3], [4], [5], [6], [7], [8], [9] are *myopic*, targeting low-cost solutions to their immediate objective without regard to the future, a pervasive planning strategy that often incurs side effects on subsequent tasks that increase overall cost.

Received 23 June 2025; accepted 19 October 2025. Date of publication 10 November 2025; date of current version 24 December 2025. This article was recommended for publication by Associate Editor H. Zhang and Editor O. Stasse upon evaluation of the reviewers' comments. This material is based upon work supported by the National Science Foundation (NSF) under Grants 2232733 and 2350352; G. J. Stein acknowledges support from this grant and from the U.S. Army Research Laboratory (W911NF2520011). The RobotiXX Lab at GMU is supported by the NSF, the U.S. Army (ARO/ARL: W911NF2320004, W911NF2420027, W911NF2520011), AFRL, AFCENT, Google DeepMind, Clearpath Robotics, Raytheon Technologies, Tangenta, the Mason Innovation Exchange, and Walmart. *(Corresponding author: Roshan Dhakal.)*

Roshan Dhakal, Duc M. Nguyen, Xuesu Xiao, and Gregory J. Stein are affiliated with the CS Department, George Mason University, Fairfax, VA 22030 USA (e-mail: rdhakal2@gmu.edu; mnguy21@gmu.edu; xiao@gmu.edu; gjstein@gmu.edu).

Tom Silver is an Assistant Professor at Princeton University, Princeton, NJ 08544 USA (e-mail: tsilver@princeton.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3631357, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3631357

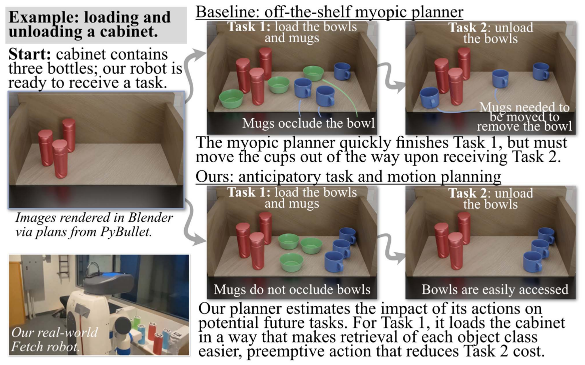
Example: loading and unloading a cabinet.
Start: cabinet contains three bottles; our robot is ready to receive a task.
Images rendered in Blender via plans from PyBullet.
Our real-world Fetch robot
Baseline: off-the-shelf myopic planner
Task 1: load the bowls and mugs
Mugs occlude the bowl
Task 2: unload the bowls
Mugs needed to be moved to remove the bowl
The myopic planner quickly finishes Task 1, but must move the cups out of the way upon receiving Task 2.
Ours: anticipatory task and motion planning
Task 1: load the bowls and mugs
Mugs do not occlude bowls
Task 2: unload the bowls
Bowls are easily accessed
Our planner estimates the impact of its actions on potential future tasks. For Task 1, it loads the cabinet in a way that makes retrieval of each object class easier, preemptive action that reduces Task 2 cost.

Fig. 1. **Anticipatory** TAMP reduces cost by anticipating how plans *now* impact tasks that may later be assigned, shown here to improve performance in our cabinet-loading scenario.

Consider the cabinet-loading scenario of Fig. 1. Myopic planning via an off-the-shelf TAMP solver [10] quickly loads the mugs and bowls into the cabinet yet in a configuration that impedes `unload the bowls`, a task the robot might next be assigned for which the mugs must be moved out of the way. If the robot were to instead *anticipate* that it may later be tasked to `unload the bowls` or `unload the mugs`, it would load the cabinet so as to avoid such side effects. As shown in Fig. 1 (bottom), this small immediate expenditure of additional effort reduces cost overall.

The cabinet-loading scenario falls within the realm of *anticipatory planning* [11], [12] an emerging subfield in which a robot jointly considers the cost of accomplishing its current task—specified as a high-level symbolic goal the robot should achieve—and the impact of its solution on subsequent tasks. As it will not know its future tasks in advance, the robot must instead plan with respect to a *task distribution*, which specifies what future tasks may later be assigned and their relative likelihood. Anticipatory planning thus involves searching over the space of plans to find the one that jointly minimizes the immediate plan cost and the *expected* future cost.

Recent work in the space of anticipatory planning [11], [12], [13], [14], so far focuses specifically on anticipatory *task* planning problems, for which the state space is discrete; one such approach [12] considers anticipation in homes, yet only over symbolic states, not considering geometric constraints and continuous motions. However, rearrangement-style tasks in general often require jointly reasoning about both discrete elements of the state (e.g., whether an object is loaded into the cabinet) and continuous parameters of the state: e.g., where inside the cabinet the object is placed. Integrated TAMP, designed to solve such problems, is inherently complex due to the interconnected nature of the discrete and the continuous, since removing an object from the back of a cabinet may first require moving other objects out of the way. This challenge is further amplified by the need to anticipate how the robot's actions may negatively impact potential future tasks.

Just as TAMP requires specialized solvers, exhibiting anticipatory behaviors in a continuous setting requires novel contributions that intertwines search over an integrated discrete-continuous state space. Existing anticipatory planning strategies are not well-suited to reason about these continuous aspects of the state. Our cabinet-loading scenario illustrates the importance of their consideration: though loading all objects inside the cabinet specifies only a single *symbolic state*, how those objects are arranged within the cabinet strongly determines how easily objects can be subsequently unloaded. This work develops an approach that, unlike the dominant paradigm, accounts for subsequent tasks to improve the performance of long-lived robots that live in an environment that persists between tasks.

We present *Anticipatory Task and Motion Planning*, which improves robot performance of over long deployments in persistent environments, each deployment a sequence of tasks assigned one-at-a-time, by imbuing them with the ability to anticipate the impact of their actions on future tasks in continuously-valued manipulation and rearrangement tasks. Difficult to compute exactly, the expected future cost is estimated via learning using a graph neural network (GNN) that consumes a graphical representation of the state. A model-based TAMP planner [5], [10] generates candidate plans, in effect sampling over the continuous goal space, and we select the plan that minimizes the total cost: (i) the immediate cost from the TAMP planner plus (ii) the estimated expected future cost from our learned estimator. Using learning and planning in tandem, our approach quickly and reliably completes the assigned objective while also producing solutions that reduce overall cost over lengthy many-task deployments.

The contributions of this work are as follows: (1) the insight that a lack of anticipation is a source of poor performance for robots operating in persistent continuous-space environments; no approach exists that can jointly reason about the continuous aspects of state and anticipation. (2) a novel approach that can address both the challenges of anticipation and TAMP: an anytime approach that leverages an off-the-shelf TAMP planner, to benefit from its reliability and generality in continuous-space settings, and uses a learned expected future cost estimator to select future-conscious continuous-space plans from the solver. (3) experimental evaluations that validate our insight and afford significant performance gains in both simulated environments and real-robot experiments that reflect elements pervasive among mobile robots.

We evaluate the performance of our learning-augmented planning approach in two domains: an object-reaching scenario in a navigation among movable obstacles (NAMO) domain and a cabinet-loading domain. We demonstrate that our approach reduces average plan cost by 32.7% over 20-task sequences in the NAMO domain and by 16.7% over 10-task sequences in the cabinet domain. Furthermore, if given time in advance to *prepare* the environment before any tasks are assigned, we demonstrate performance improvements of 83.1% and 22.3% in the NAMO and cabinet domains respectively. In both simulated and real-world experiments, we show the benefit of our learning-augmented TAMP strategy, improving performance over deployments in persistent environments consisting of multiple tasks given in sequence, a step towards more performant long-lived robots.

II. Related Work

Task and Motion Planning: TAMP involves jointly reasoning over discrete (`place the bowl`) and continuous (`at pose` X) spaces to achieve long-horizon goals (`load the bowls and mugs`) [15], [16]. We build on the sampling-based TAMP planner of Srivastava et al. [5]. Existing TAMP approaches [1], [2], [3], [4], [5], [8], [9], [17], [18] solve tasks in isolation. We show empirically that this *myopic* approach performs poorly when the environment persists and solutions to one task may impact the next.

Feasibility Checking for Planning: Lagriffoul and Andres [19] use geometric features to prune infeasible plans. Learning-based methods also exist to predict feasibility [20], [21], [22]. While these methods improve efficiency by specifically focusing on feasibility checks and predictions, our approach explicitly estimates expected future costs to avoid negative side effects in general, improving performance averaged over all possible future tasks even in the absence of any feasibility constraints or considerations.

Anticipating and Avoiding Side Effects: Recent work has explored how anticipating future tasks can improve planning: e.g., via estimation of expected future costs [11], [12], task prediction using LLMs [13], or modeling human routines to enable proactive assistance [14]. However, these methods focus on *discrete-space task planning* rather than integrated TAMP, which requires reasoning over both symbolic and continuous state spaces. Therefore, a new approach is required that can reason about anticipation in the joint discrete-continuous space of TAMP. Other works in reinforcement learning [23] and learning from demonstration [24], [25], [26], [27] aim to acquire helpful behaviors through interaction or example, and could help reduce side effects. However, their direct application is difficult in our setting, which is non-deterministic, long-horizon, and requires generalization across symbolic and geometric constraints.

Integrating Planning and Learning: To address some of the computational challenges in task planning and TAMP, recent advances have leveraged learning: learning heuristics for task-level planning [6] or sampling distributions for continuous-space planning [6], [28], [29]. Recently, the use of LLMs has also been explored in TAMP: language-guided object rearrangement [30], planner-motion feedback loops that reason over inverse kinematics/collision failures [31], as translators/checkers [32], and for grounding [33]. However, all such approaches focus on solving a single task in isolation and are also not well-suited to address the unique challenges of our anticipatory TAMP objective. Our learned model for this work is a graph neural network (GNN) [34], [35], [36], as they have proven effective for TAMP problems [9], [37], [38], [39].

III. Anticipatory Task and Motion Planning

A. Preliminaries

Task and Motion Planning: We define a task and motion planning (TAMP) problem as a tuple $\langle \mathcal{S}, \mathcal{F}, \mathcal{A}, s_0, \tau \rangle$ following the conventions of Chitnis et al. [7]. $\mathcal{S}$ defines the configuration space of the robot and all moving objects in the environment. $\mathcal{F}$ are Boolean predicates describing symbolic relationships; $\mathcal{A}$ are high-level operators such as `pick`, `move`, and `place`. The initial state is $s_0 \in \mathcal{S}$, and the task τ is a set of goal fluents defining the goal region $G_\tau \subset \mathcal{S}$.

The goal of a TAMP planner is to find a sequence of actions $\pi = [a_0, \ldots, a_n]$ such that the final state $\mathrm{tail}(\pi) \in \mathrm{G}_\tau$. We assume access to such a planner. In this work, we use the planner of Srivastava et al. [5], [10], which uses the Planning Domain Definition Language (PDDL) [40] and FastDownward [41] for symbolic task planning and leverages RRT-Connect [42] for motion planning, to refine task skeletons into feasible trajectories. For notational convenience later, we represent a state s as a tuple $\langle \sigma, k \rangle$, representing the *discrete symbolic components of the state* σ (e.g., on which surface an object is placed) and the *continuous aspects of the state* k: e.g., where on that surface the object is placed.

Anticipatory Planning: There is often considerable flexibility in how the robot can choose to complete its assigned task τ: i.e., the goal region G_τ consists of more than a single satisfying state. During long-lived deployments, the environment will persist *between tasks*, and so effective planning requires that the robot consider how its choice of how to complete the current objective—which goal state s_g it ends up in—impacts possible tasks it may later be assigned, where tasks are assigned according to a *task distribution* $P(\tau)$. We treat this distribution as static, a realistic setting in which the robot's responsibilities do not change over its lifetime. We adopt the formalism of Dhakal et al. [11], [12], which defines anticipatory planning as a joint minimization over (i) the cost to complete its assigned current task τ_c and (ii) the expected future cost to complete a subsequent task:

$$s_g^* = \underset{s_g \in G(\tau_c)}{\operatorname{argmin}} \Bigg[\overbrace{V_{s_g}(s_0)}^{\substack{\textbf{Immediate Task Cost:} \\ \text{Cost to reach } s_g \text{ from } s_0}} + \overbrace{\sum_\tau P(\tau) V_\tau(s_g)}^{\substack{\textbf{Anticipatory Planning Cost:} \\ \text{Expected cost over future tasks}}} \Bigg], \tag{1}$$

where $V_{s_g}(s_0)$ is the plan cost to reach state s_g from starting state s_0, and $V_\tau(s_g)$ is the cost to complete task τ from state s_g, and the notational shorthand $\sum_\tau$ indicates a sum with a contribution from each task τ in the task distribution $P(\tau)$. As mentioned by Dhakal et al. [11], reasoning about expected future cost over long sequences of tasks is computationally challenging, and so Eq. (1) instead seeks to minimize cost over an immediate task and a single next task in the sequence; we will show in Section V that this formulation is still sufficient for improved behavior over lengthy sequences. We note that the aim of anticipatory planning is not to directly *predict* what the next task might be, but to plan to reduce *expected* future cost.

Owing to the difficulty of integrated TAMP, recent work in this space [11], [12], [13], [14] considers only *task planning* settings, focusing only on symbolic planning and thus ignoring continuously-valued aspects of the state.

B. Problem Formulation: Anticipatory TAMP

We solve the problem of anticipatory TAMP, which combines elements of both TAMP and anticipatory planning and so is defined by the tuple: $\langle \mathcal{S}, \mathcal{F}, \mathcal{A}, s_0, \tau, P(\tau) \rangle$. Tackling this problem requires reasoning about both discrete and continuous aspects of the state and so anticipatory TAMP in general involves solving the following objective:

$$\begin{aligned} \sigma_g^*, k_g^* &= \underset{\sigma_g, k_g \in G(\tau_c)}{\operatorname{argmin}} \Bigg[\overbrace{V_{\sigma_g, k_g}(s_0)}^{\substack{\textbf{Immediate Task Cost:} \\ \text{Cost to reach } s_g \equiv \langle \sigma_g, k_g \rangle}} + \overbrace{\sum_\tau P(\tau) V_\tau(\langle \sigma_g, k_g \rangle)}^{\substack{\textbf{Anticipatory Planning Cost:} \\ \text{Expected cost over future tasks}}} \Bigg] \\ &= \underset{\sigma_g, k_g \in G(\tau_c)}{\operatorname{argmin}} \Big[V_{\sigma_g, k_g}(s_0) + \underbrace{V_{\text{A.P.}}(\langle \sigma_g, k_g \rangle)}_{V_{\text{A.P.}} \equiv \text{ expected future cost}} \Big], \end{aligned} \tag{2}$$

where $V_{\sigma,k}(s_0)$ is the plan cost to reach state $s \equiv \langle \sigma, k \rangle$ from initial state s_0 and $V_\tau(\langle \sigma_g, k_g \rangle)$ is the cost to complete task τ from state $s_g \equiv \langle \sigma_g, k_g \rangle$.

Preparation as Task-Free Anticipatory TAMP: Household robots will not be in perpetual use and so can take preemptive action to *prepare* the environment: transform or rearrange the environment to reduce expected future costs and thus make it easier to complete tasks once they are eventually assigned. Formally, preparation can be defined as *task-free anticipatory planning* [11]. For anticipatory TAMP, we define preparation as minimization over the space of continuous states k associated with the current symbolic state σ_0, $k_{prep} \in K(\sigma_0)$: e.g., rearranging the objects within the cabinet. Thus, the prepared state of the environment $\langle \sigma_0, k^*_{prep} \rangle$ is defined via:

$$k^*_{\text{prep}} = \underset{k_{\text{prep}} \in K(\sigma_0)}{\operatorname{argmin}} \left[V_{\text{A.P.}}(\langle \sigma_0, k_{\text{prep}} \rangle) \right]. \tag{3}$$

C. Approach: Planning Via Anticipatory TAMP

Our *anticipatory task and motion planning* involves selecting a plan that minimizes total cost: the sum of immediate plan cost and expected future cost, $V_{\text{A.P.}}$ (Eq. (2)). Since computing $V_{\text{A.P.}}$ online is intractable in general, we use a learned model, APCOSTESTIMATOR, to estimate it during planning. Fig. 2 and Algorithm 1 outline this process.

Our method builds on an existing randomized TAMP solver [5], [10], which produces diverse plans by sampling over the continuous goal space. Each call returns a valid plan π ending in a randomly sampled goal state. We query TAMPSOLVER N times to generate N candidate plans, estimate $V_{\text{A.P.}}$ for each using APCOSTESTIMATOR, and return the plan with the lowest total cost according to Eq. (2). This algorithmic approach is similar to that of Talukder et al. [12], who use a random sampling approach as a computation saving measure to search the space of possible states to select the state that jointly minimizes the sum of immediate cost and the anticipatory cost, consistent with the general framework of anticipatory planning. Our approach relies on a TAMP solver to search the space of plans and so is capable of continuous-space reasoning necessary for our setting.

This approach is anytime. Both the anticipatory TAMP solver and the myopic TAMP solver are run multiple times as computation allows (as would be determined by the user or setting) with a different selection criteria used to select the "best" plan. The only added cost over the baseline is the negligible overhead of the learned estimator. When computation is *not* available more than once, both approaches return the only generated plan. With more computation, our method yields clear benefits, as shown in Section V-D.

Due to the high computational cost of searching over hybrid state spaces, in this work we specifically study tasks with well-specified symbolic goals. This shifts the planning emphasis

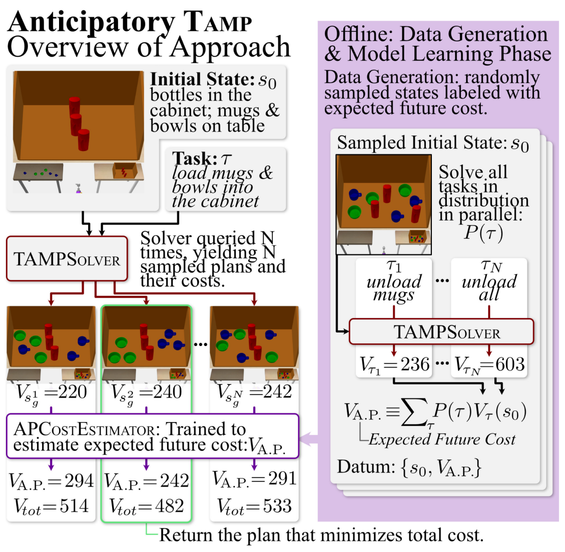
Anticipatory TAMP
Overview of Approach
Initial State: s_0 bottles in the cabinet; mugs & bowls on table
Task: τ load mugs & bowls into the cabinet
TAMPSolver
Solver queried N times, yielding N sampled plans and their costs.
$V_{s_g^1}=220$
$V_{s_g^2}=240$
$V_{s_g^N}=242$
APCostEstimator: Trained to estimate expected future cost: $V_{\text{A.P.}}$
$V_{\text{A.P.}}=294$ $V_{tot}=514$
$V_{\text{A.P.}}=242$ $V_{tot}=482$
$V_{\text{A.P.}}=291$ $V_{tot}=533$
Return the plan that minimizes total cost.
Offline: Data Generation & Model Learning Phase
Data Generation: randomly sampled states labeled with expected future cost.
Sampled Initial State: s_0
Solve all tasks in distribution in parallel: $P(\tau)$
τ_1 unload mugs
τ_N unload all
TAMPSolver
$V_{\tau_1}=236$
$V_{\tau_N}=603$
$V_{\text{A.P.}} \equiv \sum_\tau P(\tau)V_\tau(s_0)$
Expected Future Cost
Datum: $\{s_0, V_{\text{A.P.}}\}$

Fig. 2. **Schematic of our Anticipatory TAMP Approach**. As detailed in Alg. 1, a TAMP solver is queried N times and we choose the solution that minimizes the sum of immediate task planning cost and expected future cost, estimated by a learned model: APCostEstimator.

to continuous aspects—for instance, placing all objects in a cabinet defines a single symbolic goal, yet requires reasoning over their exact continuous placements. However, we note that our contributions are generally applicable.

Preparation, task-free anticipatory TAMP, searches for the state that minimizes expected future cost in advance of being given a task. We perform this search via a simulated annealing optimization approach [43]. Beginning with an initial state $\langle \sigma_0, k_0 \rangle$, we iteratively perturb the continuous object states k to find valid configurations. If the $V_{\text{A.P.}}$ of the new state is improved or meets a probabilistic criterion influenced by a decreasing temperature factor, it is accepted as the new state to perturb. Search proceeds for N iterations or as computation allows, eventually returning the prepared state $\langle \sigma_0, k^*_{\text{prep}} \rangle$.

IV. Estimating Expected Future Cost

During deployment, direct computation of the anticipatory planning cost $V_{\text{A.P.}}(\langle \sigma_g, k_g \rangle)$ is often infeasible, due to the high computational cost of solving all possible future tasks for every state considered during planning or because the robot may not have direct access to the underlying task distribution when deployed. Instead, we estimate $V_{\text{A.P.}}$ via an estimator (APCostEstimator): a GNN [34], [35] trained via supervised learning with data generated during an offline training phase.

The one-time investment to train the estimator obviates the need for difficult, exhaustive computation of expected future cost during deployment. Our anytime approach comes with only negligible computational overhead and achieves forward-looking behaviors out of reach for its myopic counterpart.

Training Data Generation Offline, we assume access to the task distribution $P(\tau)$, which defines possible tasks and their likelihoods. To generate training data, we sample states s_i from the domain of interest and solve future tasks τ using TAMPSolver, computing plan costs $V_\tau(s_i)$ to compute the anticipatory planning cost via its definition: $V_{\text{A.P.}} \equiv \sum_\tau P(\tau)V_\tau(s_i)$. Each datum consists of a state s_i with its corresponding $V_{\text{A.P.}}$, which the model learns to estimate. Ubuntu 22.04 is used with an NVIDIA RTX A4000 GPU.

Algorithm 1: Anticipatory TAMP

```
function AntTAMP($s_0, \tau$, APCostEstimator)
    $V^*_{\text{total}} \leftarrow \infty$
    for $i \in \{1, 2, \ldots, N\}$ do
        ▷Get a candidate plan and its cost for $\tau$ using off-the-shelf TAMP solver, sampling goal states.
        $\pi$, PlanCost $\leftarrow$ TAMPSolver($s_0, \tau$)
        $\langle \sigma_g, k'_g \rangle \leftarrow$ Tail($\pi$)
        ▷Myopic planning uses a zero-function in place of the $V_{\text{A.P.}}$ estimator, as it does not anticipate.
        $V_{\text{A.P.}} \leftarrow$ APCostEstimator($\langle \sigma_g, k'_g \rangle$)
        $V_{\text{total}} \leftarrow$ PlanCost $+ V_{\text{A.P.}}$
        if $V_{\text{total}} \leq V^*_{\text{total}}$ then
            $\pi^* \leftarrow \pi$,  $V^*_{\text{total}} \leftarrow V_{\text{total}}$
    return $\pi^*$
```

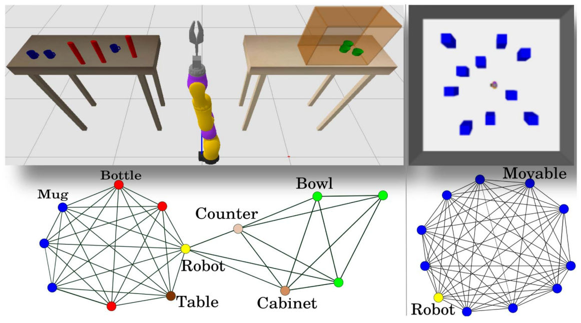

Fig. 3. Example states and their graph representations for both our Cabinet and NAMO domains for GNN training.

Learning and Estimation via Graph Neural Networks As we rely on a GNN for learning and estimation, we represent the environment state $\langle \sigma, k \rangle$ as a graph $\mathcal{G}$, as shown in Fig. 3. Nodes represent objects—e.g., the robot, movable objects, and object containers—and edges represent spatial or semantic relationships between them: e.g., that a bowl is in the cabinet or a mug is on the table, as visualized in Fig. 3. $\mathcal{G}$ includes features for both nodes and edges, specific to each environment; see details alongside our experiments in Section V. Our GNN is implemented via PyTorch Geometric [44] and consists of three TransformerConv layers [35] each followed by a leaky ReLU activation, culminating in a mean-sum pooling operation and a fully-connected layer. We use a mean absolute error loss and train with AdaGrad [45] with a batch size of 8 for 10 epochs with a 0.05 learning rate.

V. Experiments and Results

We first evaluate our approach on two PyBullet [46] simulated domains based on those by Chitnis et al. [10]: (A) object-reaching in a navigation among movable obstacles (NAMO) domain and (B) cabinet-loading.

For each trial, we evaluate performance of four planners.

Myopic Myopic TAMP, which does not anticipate future cost. Planning relies on Alg. 1 using a zero-function for the anticipatory cost estimator, ensuring fair comparison.

AntTAMP Our anticipatory TAMP approach, which seeks to minimize both immediate and expected future cost via Eq. (2). We plan via Alg. 1 using a learned anticipatory cost estimator, trained in the environment of interest.

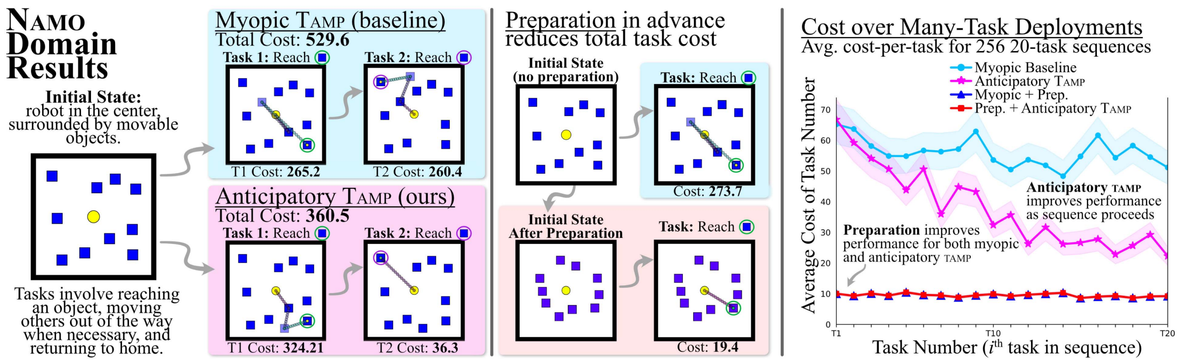

Fig. 4. **Navigation among movable obstacles (NAMO) Domain Results**. Left: In examples, ANTTAMP moves obstacles so that they are out of the way, reducing cost for subsequent tasks. This improves upon the MYOPIC plan, which cannot consider potential side effects. Given time to *prepare*, our robot rearranges the objects into a ring-like structure that ensures all objects are easily reached without any `moveclear` actions. Visualized plans omit the robot's return–to–home for clarity. Right: the average per-task performance over time (i.e., as each sequence proceeds) shows that anticipatory TAMP tends towards reducing average task cost over its deployment. Cost is further reduced by preparation.

PREP+MYOPIC The robot first *prepares* the scene via Eq. (3) (Section III-B) and then plans via MYOPIC.

PREP+ANTTAMP The robot first *prepares* via Eq. (3) and then plans via ANTTAMP.

Our central aim is to show the need for anticipation in a continuous-space setting, validating our insight that without our ANTTAMP approach, the robot will not achieve these desirable behaviors. Thus, the commonly-used MYOPIC planning is the most appropriate baseline for our ANTTAMP approach.

A. Object Reaching Scenario in a NAMO Domain

We first evaluate in a navigation among movable obstacles (NAMO) domain, where tasks require that the robot navigate to a target object, as if the object requires interaction or inspection, and then return to the start. This standard TAMP setup, extending the domain from [10], reflects cluttered home and warehouse domains (as in [10], [39]). The task distribution is uniform over 10 objects. `moveclear` actions are needed to move blocks that obstruct traversal, incurring a base cost of 200 plus Euclidean distance for movement. Reaching the target object may require moving other objects out of the way, and so good performance requires that the robot position objects so that they do not block others.

Our anticipatory cost estimator GNN is trained on 10K data, each labeled with expected future cost approximated by sampling from the distribution of future tasks analytically-computed expected future cost, an expensive operation taking roughly 10 minutes per labeled example, prohibitive to do during planning. Training with these generated samples takes approximately five minutes. Each graph node represents an entity (robot or object) with input features: one-hot type encoding, pose, and distance to the robot. Edges encode pairwise distances and count the number of obstructing objects the straight-line edge touches. We evaluate our approach on 256 trials, where each trial consists of a random sequence of 20 tasks. For each task, Alg. 1 generates 100 candidate plans (see Section V-D), requiring approximately 1–2 minutes of planning time per task, and with 5000 iterations for preparation.

Table 1 reports the average cost-per-task across four planning strategies. All trials succeed using the model-based TAMP solver. Compared to MYOPIC, ANTTAMP reduces planning cost by 32.7%. Preparation further improves performance, with PREP+ANTTAMP and PREP+MYOPIC achieving identical 83.1% lower costs than MYOPIC: preparation consistently finds states from which all objects are accessible without rearrangement, thus, anticipatory TAMP does not need to go out of its way to rearrange the blocks during task execution, as doing so would increase overall cost.

TABLE I
AVERAGE COST-PER-TASK OVER 20-TASK SEQUENCES ACROSS PLANNING APPROACHES IN THE NAMO AND CABINET DOMAINS. *THE NAMO 11–13 AND CABINET 10 DOMAINS INCLUDE ADDITIONAL OBJECTS FOR EVALUATING LIMITED TEST-TIME SHIFT; SEE SECTION V-C

Planning Approach	NAMO	Cabinet	NAMO 11–13*	Cabinet 10*
MYOPIC	56.1	283.2	108.3	314.8
ANTTAMP (ours)	**37.8**	**235.8**	**92.5**	**281.5**
PREP (ours)+MYOPIC	**9.5**	267.7	48.9	303.1
PREP+ANTTAMP (ours)	**9.5**	**219.9**	**45.8**	**272.7**

Fig. 4 (right) shows that ANTTAMP lowers per-task cost over time—indicating that the environment becomes easier to navigate through repeated interaction, an emergent benefit of anticipatory planning. We highlight a few examples in Fig. 4 that corroborate the quantitative results, namely that planning via ANTTAMP ensures that obstacles are placed so as to be out of the way for subsequent tasks. Prepared states often exhibit a *ring-like* structure that allows unhindered access to all objects.

B. Cabinet Loading and Unloading Scenario

Our cabinet domain consists of nine objects: three each of mugs (blue), bottles (red), and bowls (green), using URDF models from Liu et al. [47]. Each task involves loading or unloading all objects of one or more semantic classes—e.g., move all bottles to the table—each assigned with equal probability. The domain includes three operators to complete a task: `move` (with cost based on Euclidean distance), and `pick` and `place` (each with a fixed cost of 20 units).

To estimate the anticipatory planning cost, we train a GNN on 5000 states labeled with the expected future cost. It takes roughly twenty minutes to label each datum and approximately eight minutes to train the model. Node features include a one-hot encoding of entity type (robot, container, or object), pose, and

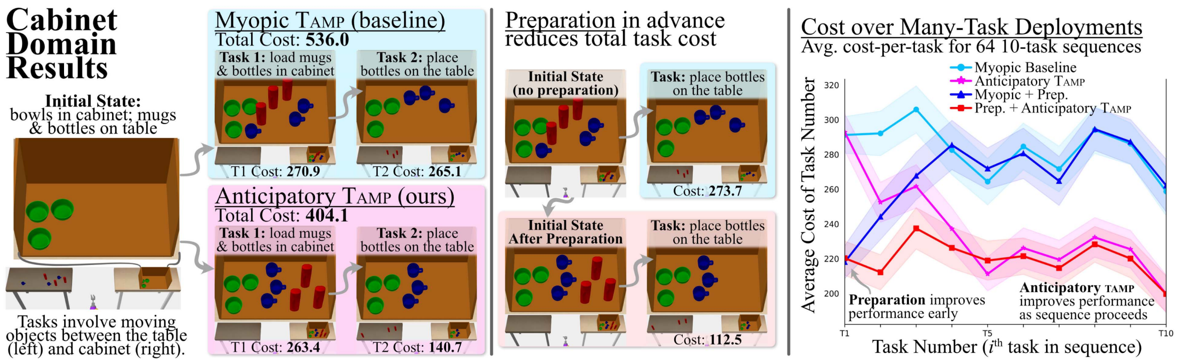

Fig. 5. **Cabinet Domain Results.** Left: MYOPIC completes the *load the cabinet* task in a way that causes obstructions when the robot must later *unload the bottles*. Instead, ANTTAMP groups similar objects when loading, making it easier to unload later, reducing overall planning cost. *Preparation* similarly organizes the cabinet so that subsequent tasks are made easier, showing the strength of our approach. Right: the average per-task performance over time (i.e., as the sequence proceeds) shows that ANTTAMP tends towards improved average task cost over its deployment while MYOPIC tends towards poor performance, even with preparation in advance.

distance to the robot. Edges represent spatial and semantic relationships, with features including inter-node distance and the number of movable obstacles (set to zero for non-movable objects). Objects of the same semantic class are interchangeable, so only objects from other classes are treated as obstacles when computing edge features. We evaluate our system across 64 deployments, where each deployment consists of a sequence of 10 tasks. For each task, Alg 1 samples 200 candidate plans, with planning taking approximately 2–4 minutes per task. We use 2500 candidate states for preparation.

Table 1 shows the average cost-per-task for four planning strategies. Since our approach relies on model-based TAMP, all trials succeed in completing the assigned task for all planners. ANTTAMP reduces cumulative planning cost by 16.7% compared to MYOPIC. Preparation further lowers cost for both, yielding a 22.3% improvement over MYOPIC.

We highlight the performance-over-time in Fig. 5 (right), which shows that ANTTAMP increasingly outperforms MYOPIC over time. This encouraging result shows that as our robot continues to be given tasks *the environment becomes easier and easier to use*, behavior that leads to cabinet configurations that increasingly reduce expected cost: *organizing* the cabinet without explicit examples of how the cabinet should be laid out. Fig. 5 (left) show example cabinet layouts during planning via the various strategies.

We can see similar performance gains afforded by *preparation*, by which the environment is rearranged in advance to reduce expected future cost. Preparation improves the performance of both planning strategies, yet over time its benefits for the MYOPIC planner diminish; just as ANTTAMP gradually results in lower-expected-cost configurations of the environment, under MYOPIC planning the environment gradually tends towards higher costs as the side effects of myopic action accrue. Combining preparation with ANTTAMP yields the best results.

Fig. 5 shows example trials, which support our statistical results; each shows how planners that rely on our anticipatory TAMP have discovered the benefit of loading objects so as to avoid obstructing those of different semantic class and so tend to reduce overall cost over the course of each 20-task sequence. Moreover, though our approach only involves direct optimization with respect to a single subsequent task, our approach yields improvements that tend towards lower expected cost over the course of multiple tasks, as can be seen in the ANTTAMP results in Fig. 5, emergent behavior that further reinforces the benefit of our approach.

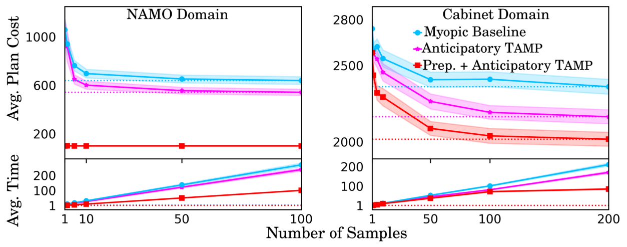

Fig. 6. Average cumulative task cost and average planning time (in seconds) versus number of samples for ANTTAMP and MYOPIC over 64 and 128 trials on Cabinet and NAMO domains, respectively. Anticipatory TAMP improves performance across varied effort.

C. Evaluation Under Limited Test-Time Shift

We conduct additional experiments in each study to study performance under limited shifts at test-time. We add additional objects—NAMO 11–13 contains an additional 1–3 movable objects and Cabinet 10 contains an additional bowl or mug or bottle in the cabinet domain—and shift the task distribution to accompany these changes: e.g., to *unload 4 mugs from the cabinet*, a task not seen during training. Though the costs across all experiments are higher due to the increased difficulty of each new setting, the results in Table 1 still show performance improvements from our approach.

D. Performance as a Function of Computation

Our approach is *anytime*, and so performance improves with additional computation. We evaluate with 1, 2, 5, 10, and 100 samples in the NAMO domain, and up to 200 in the Cabinet domain and show performance results in Fig. 6. As expected, more samples lead to better performance for all planners. Moreover, since anticipation tends towards states in which tasks are easier to complete, Fig. 6 (bottom) additionally shows that the per-task planning time is comparatively less for our approach, a further benefit of using our approach.

Notably, even with relatively few samples, our anticipatory planner achieves meaningful reductions in the cumulative plan cost and planning time, demonstrating its effectiveness even under limited computational resources.

E. Real-World Demonstration on a Mobile Manipulator

We further evaluate our approach using the Fetch Mobile Manipulator [48] in a real-world cabinet-loading scenario with

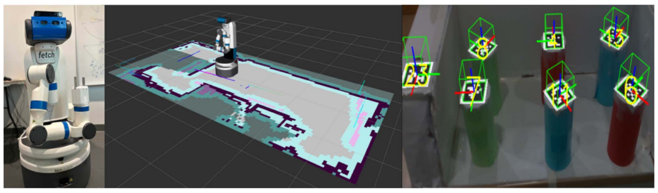

Fig. 7. Left: Our Fetch robot. Middle: Environment occupancy map. Right: April tags for object pose estimation.

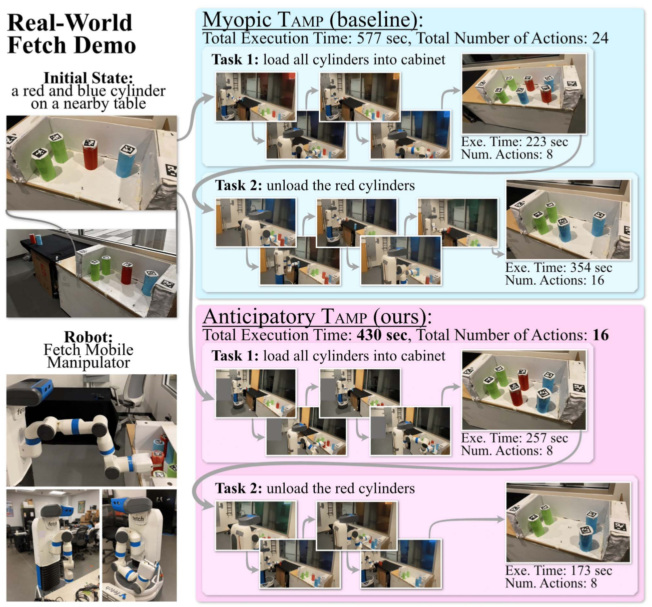
Real-World Fetch Demo
Initial State: a red and blue cylinder on a nearby table
Robot: Fetch Mobile Manipulator
Myopic TAMP (baseline):
Total Execution Time: 577 sec, Total Number of Actions: 24
Task 1: load all cylinders into cabinet
Exe. Time: 223 sec
Num. Actions: 8
Task 2: unload the red cylinders
Exe. Time: 354 sec
Num. Actions: 16
Anticipatory TAMP (ours):
Total Execution Time: 430 sec, Total Number of Actions: 16
Task 1: load all cylinders into cabinet
Exe. Time: 257 sec
Num. Actions: 8
Task 2: unload the red cylinders
Exe. Time: 173 sec
Num. Actions: 8

Fig. 8. **Real-world demonstration with the Fetch.** An example 2-task sequence (loading and unloading cylinders) given one-at-a-time, such that latter tasks are *unknown in advance*. MYOPIC takes 577 seconds (24 actions) to complete the sequence whereas our approach ANTTAMP completes them in only 430 seconds (16 actions).

six cylindrical objects—two each in red, blue, and green—and with an anticipatory cost estimator trained in simulation environments of the same composition. We demonstrate our approach on a Fetch Mobile Manipulator (Fig. 7). The robot uses the ROS `move_base` package [49] for navigation, AprilTags [50] for object pose estimation, and MoveIt [51] and IKFast [49], [51] for manipulation. The demonstrations involve first generating a plan in the simulated environments using the TAMP solver described above and then executing that plan, by sequential execution of the prescribed actions, using the aforementioned planning and perceptual modules.

We show the performance of both ANTTAMP and MYOPIC planners; both are initialized with the same starting configuration (Fig. 8) and tasked to load the two objects from the table into the cabinet. Upon completion of that first task, each robot is then instructed to unload the two red cylinders and move them to the table, a task made difficult if either red cylinder is obstructed other non-red cylinders.

The MYOPIC planner (Fig. 8, top), incapable of considering the effect of its actions on possible future tasks, solves the first task such that the back red cylinder is more difficult to remove when the second task is eventually assigned. Conversely, our ANTTAMP approach places the cylinders in the same-colored groups and so the second task is made easier. Our approach requires fewer total actions (pick, move, place) and correspondingly less total execution time to complete the tasks in the sequence, as illustrated in Fig. 8.

VI. Discussion, Limitations, and Future Work

We present anticipatory task and motion planning (TAMP), a strategy for improving performance across task sequences in persistent, continuous-space rearrangement-style settings. Unlike most existing TAMP approaches that plan one task at a time, our method uses a learned model to estimate how current actions impact future tasks, guiding the planner to balance immediate plan cost with reduction of expected future costs. In both simulated and real-world experiments, we show the benefit of our learning-augmented TAMP strategy, improving performance over deployments in persistent environments consisting of multiple tasks given in sequence, an important step towards more performant long-lived robots.

Limitations: Our approach, which relies on existing solvers during planning, suffers from many of the same challenges of scale and computation that limit TAMP for rearrangement problems in general. Moreover, planning via Algorithm 1 requires repeated TAMP solver calls to search the continuous goal space, limiting scalability to complex problems without better sampling or search. Our experiments so far only consider the (realistic) setting where a robot's responsibilities are static over its lifetime, and so deeper evaluation of the learned model may be necessary to consider scenarios where this distribution evolves or differs significantly at deployment.

In future work, we are excited to scale our approach to more complex environments, such as households, where object states (e.g., clean vs. dirty dishes) affect future cost, and task distributions may shift over time. This will require transferring knowledge from simulation or learning online as user needs evolve. Future work may also simultaneously leverage the expected future cost estimator more deeply to also guide search at the symbolic level, though would also require an investigation of how to select planning heuristics so as to ensure it does not add considerable additional computation.

References

[1] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2020, vol. 30, pp. 440–448.

[2] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Proc. Int. Conf. Robot. Automat.*, 2010, pp. 5002–5008.

[3] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1930–1936.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1470–1477.

[5] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 639–646.

[6] 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.

[7] R. Chitnis et al., "Guided search for task and motion plans using learned heuristics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 447–454.

[8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, “Incremental task and motion planning: A constraint-based approach,” in *Robot.: Sci. Syst.*, vol. 12, 2016.

[9] 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.

[10] R. Chitnis, T. Silver, B. Kim, L. Kaelbling, and T. Lozano-Perez, “CAMPs: Learning context-specific abstractions for efficient planning in factored MDPs,” in *Proc. Conf. Robot Learn.*, 2021, pp. 64–79.

[11] R. Dhakal, M. R. H. Talukder, and G. J. Stein, “Anticipatory planning: Improving long-lived planning by estimating expected cost of future tasks,” in *Proc. Int. Conf. Robot. Automat.*, 2023, pp. 11538–11545.

[12] M. R. H. Talukder, R. I. Arnob, and G. J. Stein, “Anticipatory planning for performant long-lived robot in large-scale home-like environments,” in *Proc. Int. Conf. Robot. Automat.*, 2025, pp. 10831–10837.

[13] R. Arora et al., “Anticipate & act: Integrating LLMs and classical planning for efficient task execution in household environments,” *in Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 14038–14045.

[14] M. Patel and S. Chernova, “Proactive robot assistance via spatio-temporal object modeling,” in *Proc. Conf. Robot Learn.*, 2022, pp. 881–891.

[15] C. R. Garrett et al., “Integrated task and motion planning,” *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, pp. 265–293 2021.

[16] Z. Zhao et al., “A survey of optimization-based task and motion planning: From classical to learning approaches,” *IEEE/ASME Trans. Mechatronics*, vol. 30, no. 4, pp. 2799–2825, 2025, doi: 10.1109/TMECH.2024.3452509.

[17] A. Curtis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. Kaelbling, “Discovering state and action abstractions for generalized task and motion planning,” in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 5, pp. 5377–5384.

[18] C. P. Bradley and N. Roy, “Learning to guide search in long-horizon task and motion planning,” in *Proc. CoRL 2022 Workshop Learn., Percep., Abstraction Long- Horiz. Plan.*, 2022.

[19] 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.

[20] D. Driess, J.-S. Ha, and M. Toussaint, “Deep Visual Reasoning: Learning to predict action sequences for task and motion planning from an initial scene image,” in *Proc. Robot.: Sci. Syst.*, 2020.

[21] Z. Yang, C. R. Garrett, T. Lozano-Perez, L. Kaelbling, and D. Fox, “Sequence-based plan feasibility prediction for efficient task and motion planning,” in *Proc. Robot.: Sci. Syst.*, 2023.

[22] 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, pp. 1255–1262, 2019.

[23] R. Shah, D. Krasheninnikov, J. Alexander, P. Abbeel, and A. Dragan, “The implicit preference information in an initial state,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[24] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annu. Rev. Control, Robot., Auton. Auton. Syst.*, vol. 3, pp. 297–330, 2020.

[25] S. Manschitz, J. Kober, M. Gienger, and J. Peters, “Learning to sequence movement primitives from demonstrations,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2014, pp. 4414–4421.

[26] D. Whitney, E. Rosen, and S. Tellex, “Learning from crowdsourced virtual reality demonstrations,” in *Proc. Int. Workshop Virtual, Augmented, Mixed Reality HRI (VAM-HRI)*, 2018.

[27] C. Moro, G. Nejat, and A. Mihailidis, “Learning and personalizing socially assistive robot behaviors to aid with activities of daily living,” *Trans. Hum.-Robot Interact.*, vol. 7, pp. 1–25, 2018.

[28] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, “Active model learning and diverse action sampling for task and motion planning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4107–4114.

[29] R. Chitnis, L. P. Kaelbling, and T. Lozano-Pérez, “Learning quickly to plan quickly using modular meta-learning,” in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 7865–7871.

[30] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, “Task and motion planning with large language models for object rearrangement,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2023, pp. 2086–2092.

[31] S. Wang et al., “LLM^3: Large language model-based task and motion planning with motion failure reasoning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 12086–12092.

[32] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, “AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 6695–6702.

[33] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, “Grounding large language models in interactive environments with online reinforcement learning,” in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 3676–3713.

[34] P. W. Battaglia et al., “Relational inductive biases, deep learning, and graph networks,” 2018, *arXiv:1806.01261*.

[35] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, “Masked label prediction: Unified message passing model for semi-supervised classification,” in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Z.-H. Zhou, Ed., Aug. 2021, pp. 1548–1554, doi: 10.24963/ijcai.2021/214.

[36] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *NeurIPS*, vol. 30, pp. 1025–1035, 2017.

[37] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, “Planning with learned object importance in large problem instances using graph neural networks,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 11962–11971.

[38] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, “Learning symbolic operators for task and motion planning,” in *Proc. Int. Conf. Intell. Robots Syst.*, 2021, pp. 3182–3189.

[39] 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.

[40] M. Fox and D. Long, “PDDL2.1: An extension to PDDL for expressing temporal planning domains,” *J. Artif. Intell. Res.*, vol. 20, pp. 61–124, 2003.

[41] M. Helmert, “The fast downward planning system,” *J. Artif. Intell. Res.*, vol. 26, pp. 191–246, 2006.

[42] J. J. Kuffner and S. M. LaValle, “RRT-Connect: An efficient approach to single-query path planning,” in *Proc. Millennium Conf., Int. Conf. Robot. Automat.*, 2000, pp. 995–1001.

[43] D. Bertsimas and J. Tsitsiklis, “Simulated annealing,” *Stat. Sci.*, vol. 8, no. 1, pp. 10–15, 1993.

[44] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” in *Proc. ICLR Workshop Representation Learn. Graphs Manifolds*, 2019.

[45] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, 2011.

[46] E. Coumans and Y. Bai, “PyBullet, a Python module for physics simulation for games, robotics and machine learning,” 2016.

[47] Z. Liu et al., “OCRTOC: A cloud-based competition and benchmark for robotic grasping and manipulation,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 486–493, Jan. 2022.

[48] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, “Fetch and freight: Standard platforms for service robot applications,” 2023.

[49] M. Quigley et al., “ROS: An open-source robot operating system,” in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, 2009.

[50] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” in *Proc. Int. Conf. Robot. Automat.*, 2011, pp. 3400–3407.

[51] D. T. Coleman, I. A. Sucan, S. Chitta, and N. Correll, “Reducing the barrier to entry of complex robotic software: A MoveIt! case study,” *J. Softw. Eng. Robot.*, vol. 5, pp. 3–16, 2014.