Anticipation as a Prior for Efficient Sampling in Task and Motion Planning

Roshan Dhakal, Abhishek Paudel and Gregory J. Stein

***Abstract*— We show that *anticipatory planning cost*—a learned estimate of the expected cost of *future* tasks from a state, originally introduced to learn the impact of an action on the future tasks which are not yet assigned—can be used as a *computational prior* that biases sampling inside an off-the-shelf task and motion planning (TAMP) solver. Across 256 trials in a simulated kitchen it reduces planning time by up to 34%, and search evaluations by 53%.**

I. Introduction

A TAMP solver searches over symbolic action sequences while sampling continuous parameters such as grasps, placements, and robot configurations [1, 2]. Much of the computational cost arises because the solver has little information about which continuous samples are worth attempting. Many samples, e.g. object poses, are geometrically infeasible and do not yield executable actions. More subtly, a sample may be feasible for the current action yet leave the environment in a state that makes another action of the same plan harder to execute. For example, in the cabinet-loading problem of Fig. 1, an off-the-shelf solver [2, 3] may repeatedly sample placements for `bowl1` that are blocked by `spoon2`. Such failures are not immediately available as symbolic conditions: the solver redraws poses and rechecks collision and reachability until its sampling budget is exhausted, only then lifting the last attempted sample's collision into the symbolic layer as a newly discovered fact, `obstruction(spoon2,bowl1)`, and finally replanning with an additional action *Replace spoon2*.

Because refinement commits to a sample as soon as it is collision-free, this coupling runs forward through the plan. A placement chosen for an early action silently shrinks the feasible set of every later action, so those actions exhaust their budgets more often, and each exhaustion adds a fact, a replan, and a fresh round of sampling. Failures thus cascade from decisions the solver has already committed to and will not revisit. Prior work reduces sampling cost with learned estimates of *feasibility*. Chitnis et al. [4] learn distributions over continuous parameters that favor values likely to admit collision-free trajectories, while PIGINet [5] predicts whether a task plan will admit motion refinements and prioritizes those that will. Neither ranks refinements that are all feasible now by how much room they leave for the rest of the plan.

In this work, we learn an estimate of *anticipatory planning cost* [6–8], the expected cost of likely future tasks from a given state, and use it as a task-agnostic computational prior within a single TAMP problem. As illustrated in Fig. 1, the estimator is trained offline by solving likely future tasks from sampled states. During planning, we draw N sets of continuous parameters, rank the resulting states by their estimated anticipatory costs, and attempt lower-cost candidates first. The approach changes only the order in which sampled parameters are attempted, leaving the planner's objective and feasibility checks unchanged and recovering the baseline exactly when the sampling budget $N{=}1$.

Roshan Dhakal, Abhishek Paudel and Gregory J. Stein are affiliated with the Department of Computer Science, George Mason University, Fairfax, VA. Email: {rdhakal2, apaudel4, gjstein}@gmu.edu

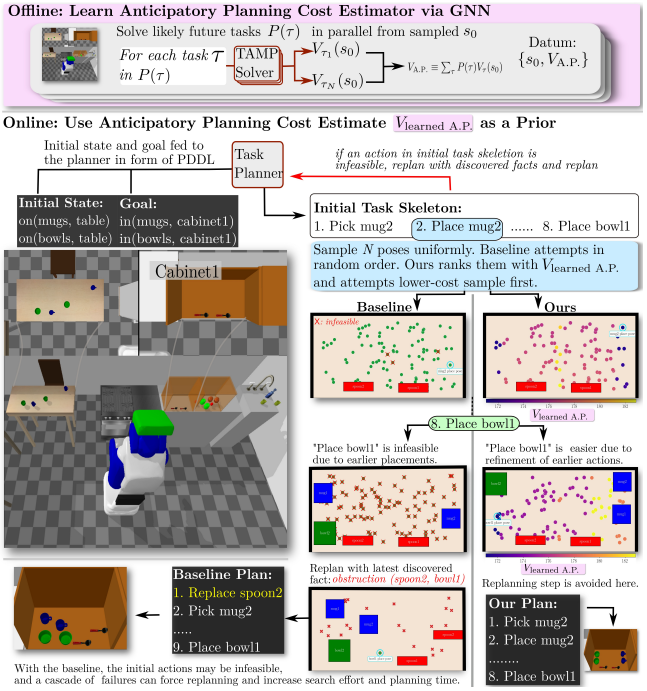

Fig. 1: Anticipatory Cost as a Prior. Offline, we learn anticipatory planning cost from likely future tasks. During planning, sampled continuous parameters are ranked by their corresponding anticipatory costs and attempted in increasing order.

II. Anticipation as a Sampling Prior

We use an existing TAMP solver [2, 3]: symbolic search with Fast Downward [9] proposes an action sequence, motion-level refinement samples continuous parameters per action, and geometric failures are lifted into new symbolic facts Φ that update the problem for next task planning process. The planner returns a plan π reaching the goal region G_τ for the task τ.

Anticipatory Cost The anticipatory cost of a state s is the expected cost of solving future tasks from it,

$$V^*_{\text{A.P.}}(s) = \sum_{\tau'} P(\tau') V^*_{\tau'}(s) \tag{1}$$

Algorithm 1 TAMP with an Anticipatory Sampling Prior. The highlighted line is the only change to the base solver.

```
function TAMP+PRIOR(s_0, τ)
    Φ ← ∅
    repeat
        π_task ← TASKPLAN(s_0, τ, Φ);   s ← s_0; π ← ∅
        for all a ∈ π_task do
            ⟨s, k, ok⟩ ← REFINE(s, a)
            if not ok then
                Φ ← Φ ∪ FAILUREFACTS; go to TASKPLAN
            π ← π ⊕ ⟨a, k⟩
    until s ∈ G_τ
    return π

function REFINE(s, a)
    for b ← 1 to B do                          ▷ B batches of N
        K ← SAMPLEPARAMS(s, a, N)              ▷ same base sampler
        if USEPRIOR then
            K ← SORTASCENDING(K, V_A.P.)
        for all k ∈ K do
            s', e ← SIMULATE(s, a, k)
            if e = ∅ then
                return ⟨s', k, true⟩
    return ⟨s, ⊥, false⟩
```

with $P(\tau')$ a task distribution and $V^*_{\tau'}(s)$ the cost of solving τ' from s; low $V_{\text{A.P.}}$ marks states from which future tasks are cheap. Computing Eq. (1) exactly is intractable, so we learn a graph neural network (GNN) predictor $V_{\text{A.P.}}(s)$ over the scene graph, trained on states labeled by Eq. (1) with future tasks sampled from $P(\tau')$ and solved offline.

Sampling Prior For each action a with continuous parameters, the base sampler proposes N candidates $\mathcal{K} = \{k_1, \ldots, k_N\}$, each yielding a state s_i. Instead of testing them in draw order, we score every candidate with a learned *anticipatory cost* $V_{\text{A.P.}}(s_i)$ and test them in increasing cost. The first feasible candidate is committed; if all N fail we draw and re-score a fresh batch of N, up to a refinement budget, after which the failure is added to Φ and the task is replanned (Alg. 1). The uniform baseline is the same procedure with the prior disabled (USEPRIOR=*false*): each batch of N is drawn from the same base sampler and tested in draw order under the same refinement budget. The two methods are thus matched in sampler, batch size N, and budget, and the prior's only effect is to rank each batch.

III. Experiments and Results

We evaluate in a simulated kitchen in PyBullet inspired by PIGINet [5]: a dining table, a counter with cabinet bins, a sink, and a stove. We compare two variants of the *same* solver over 256 trials that are identical except for the highlighted line of Alg. 1: both draw the same N candidates and commit the first collision-free one, differing only in whether those candidates are ordered by the prior. TAMP+UNIFORM tries them in draw order; TAMP+ANTPRIOR tries them in $V_{\text{A.P.}}$ order.

At a matched sampling budget, the prior consistently reaches the goal with less search evaluations and less planning time than uniform sampling (Table I), improving symbolic-search efficiency by up to 53% and planning time by up to 34%. The improvement grows with the budget to a sweet spot near $N \approx 50$: a larger pool gives the prior a better placement to select, while beyond it, the growing cost of querying the GNN offsets the search it saves. At N=1, both methods incur identical search evaluations, the residual time gap being the GNN querying time itself. Crucially, the gains appear only once the anticipatory prior has candidates to rank, confirming that they come from the ordering rather than from any change to the sampler.

TABLE I: Comparison across 256 trials. At each candidate budget N, both methods draw N candidates; Uniform retains their original order, whereas AntPrior ranks them by $V_{\text{A.P.}}$. The adjacent violins show the trial distributions for N, with vertical bars marking arithmetic means. Lower values indicate better performance.

	Plan Time (s)			Search Evaluations		
N	Uniform	AntPrior		Uniform	AntPrior	
1	**33.7**	38.4		**623**	**623**	
5	**32.9**	33.7		747	**723**	
10	36.4	**28.0**		881	**619**	
20	35.9	**24.9**		857	**400**	
50	33.9	**22.3**		598	**327**	
100	32.9	**23.8**		534	**314**	

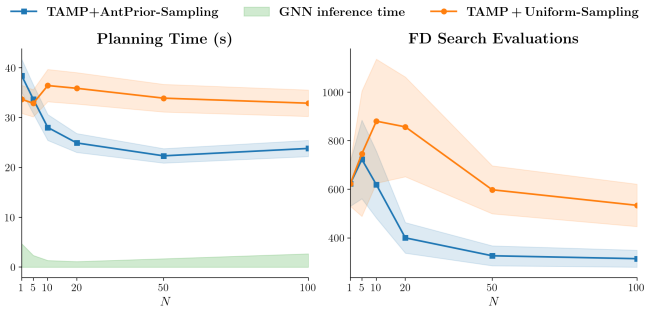

Fig. 2: Effect of the sampling budget. Both planning time and search evaluations with TAMP+ANTPRIOR are reduced compared to TAMP+UNIFORM.

IV. Conclusion

We demonstrate that our learned anticipatory planning cost is an effective computational prior for TAMP: it substantially reduces symbolic search and planning time with no operator-specific samplers, feasibility classifiers, or change to the planning objective.

References

[1] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, 2021.

[2] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[3] R. Chitnis, T. Silver, B. Kim, L. Kaelbling, and T. Lozano-Pérez, "CAMPs: Learning context-specific abstractions for efficient planning in factored MDPs," in *Conference on Robot Learning (CoRL)*, 2021.

[4] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, "Guided search for task and motion plans using learned heuristics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[5] Z. Yang, C. R. Garrett, T. Lozano-Pérez, L. Kaelbling, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[6] R. Dhakal, M. R. H. Talukder, and G. J. Stein, "Anticipatory planning: Improving long-lived planning by estimating expected cost of future tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[7] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[8] R. Dhakal, D. M. Nguyen, T. Silver, X. Xiao, and G. J. Stein, "Anticipatory task and motion planning: Improved rearrangement in persistent continuous-space environments," *IEEE Robotics and Automation Letters*, 2026.

[9] M. Helmert, "The Fast Downward planning system," *Journal of Artificial Intelligence Research*, 2006.