

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

Roshan Dhakal, Abhishek Paudel and Gregory J. Stein

Abstract—A task and motion planning (TAMP) solver spends much of its computation sampling continuous parameters, such as placements, with little information about which samples are worth attempting. A sample that is feasible for the current action can still leave the scene in a state that makes later actions of the same plan harder to refine, triggering failures, replanning, and additional search. 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 future tasks that are not yet assigned—can be used as a *computational prior* that biases sampling inside an off-the-shelf TAMP solver. Sampled candidates are ranked by their anticipatory cost and attempted in increasing order, leaving the planner’s sampler, feasibility checks, and objective unchanged. Across 256 trials in a simulated kitchen, at a matched sampling budget, the prior reduces planning time by up to 34% and search evaluations by up to 53% relative to the same solver with uniform sampling.

I. INTRODUCTION

A TAMP solver searches over symbolic action sequences while sampling continuous parameters such as grasps, placements, and robot configurations [1–5]. 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 [5, 6] 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 re-checks 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 replanning with the additional action *Replace spoon2*. The resulting replan is more expensive than the original skeleton: once `spoon2` becomes relevant to the task, the symbolic planner must reason over a larger set of reachable states and action sequences, and must additionally generate and evaluate additional continuous parameters.

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 reduces the feasible set of every subsequent action, so those actions

Roshan Dhakal, Abhishek Paudel and Gregory J. Stein are affiliated with the Department of Computer Science, George Mason University, Fairfax, VA. Email: {rdhakal2, apaude14, gjstein}@gmu.edu

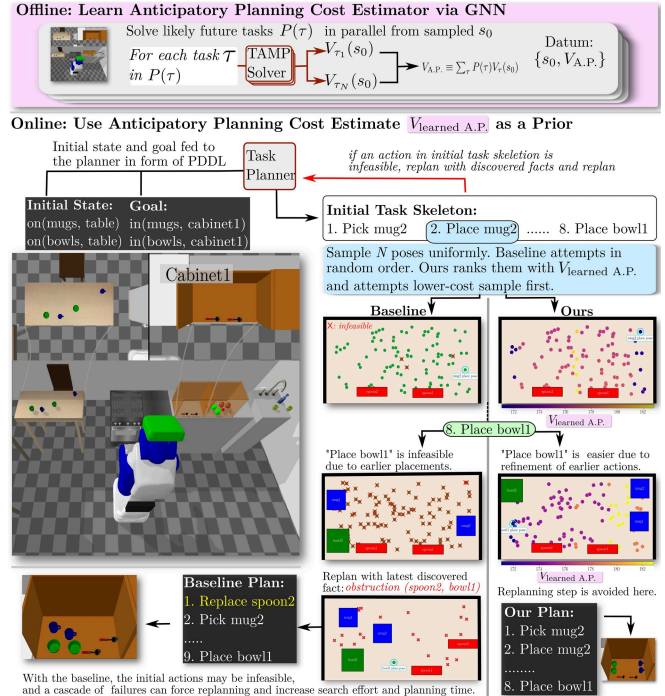


Fig. 1: Anticipatory Cost as a Prior. Offline, we learn anticipatory planning cost from likely future tasks. During planning, sampled continuous parameters are ranked by their corresponding anticipatory costs and attempted in increasing order.

exhaust their budgets more often; each exhaustion adds a fact, a replan, and a fresh round of sampling. Failures can therefore 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. [7] learn distributions over continuous parameters that favor values likely to admit collision-free trajectories, while PIGNet [8] predicts whether a task plan will admit motion refinements and prioritizes those that will. These signals are useful for avoiding candidates that are unlikely to work, but they do not rank 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* [9–11], 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 $N=1$. Because the cost is defined over states rather than operators, a single learned model scores the candidates of every action, without operator-specific samplers or feasibility classifiers. Fig. 1 contrasts the two behaviors. When refining *Place mug2*, the baseline attempts its N uniformly drawn poses in draw order, whereas our planner attempts the lower-cost poses first. The placements chosen this way leave usable space in *cabinet1*, so *Place bowl1* later refines without exhausting its budget: no obstruction fact is generated, the replanning step is avoided, and the final plan contains no rearrangement action.

We evaluate the approach in a simulated kitchen domain inspired by PIGINet [8], comparing two variants of the same TAMP solver [5, 6] that differ only in whether sampled candidates are ranked by the prior. Across 256 trials and at a matched sampling budget, the prior reduces planning time by up to 34% and symbolic search evaluations by up to 53%, with the largest savings at moderate budgets where the search saved outweighs the cost of querying the learned model.

II. RELATED WORK

TAMP and Learning informed Planning A large body of work combines symbolic planning with continuous feasibility through task-level search, sampling, refinement, or mixed discrete–continuous reasoning [1–5, 12]. Learning has been used to bias continuous parameters toward feasible values [7, 13–16], to predict the feasibility of actions or task plans so the planner can prune or reorder candidates [8, 17, 18], and to guide symbolic search directly [19–22]. These methods learn signals tied primarily to feasibility or cost for the current problem. We instead reuse a learned anticipatory cost estimate as a lightweight ranking prior over sampled parameters, which requires no operator-specific samplers or feasibility classifiers and leaves the underlying planning objective unchanged.

Anticipatory Planning Anticipatory planning is formalized as minimizing the immediate task cost plus the *expected* cost of future tasks, and show that a learned estimator can augment classical search to reduce long-term cost in symbolic-level planning [9, 10], and also have been applied to persistent continuous-space rearrangement in TAMP settings [11]. In parallel, task-anticipation approaches use LLMs to propose likely upcoming tasks and couple these predictions with classical planners to improve efficiency over sequential executions [23]. In contrast, we use anticipatory cost as a computational prior that changes only *how* a single TAMP problem is solved, not the objective being considered.

III. ANTICIPATION AS PRIOR FOR TASK AND MOTION PLANNING

A. Preliminaries

We use an existing TAMP solver [5, 6]: symbolic search with Fast Downward [24] using Planning Domain Definition

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

```

function TAMP+PRIOR( $s_0, \tau$ )
   $\Phi \leftarrow \emptyset$ 
  repeat
     $\pi_{\text{task}} \leftarrow \text{TASKPLAN}(s_0, \tau, \Phi); \quad s \leftarrow s_0; \pi \leftarrow \emptyset$ 
    for all  $a \in \pi_{\text{task}}$  do
       $\langle s, k, ok \rangle \leftarrow \text{REFINE}(s, a)$ 
      if not  $ok$  then
         $\Phi \leftarrow \Phi \cup \text{FAILUREFACTS}; \text{go to TASKPLAN}$ 
       $\pi \leftarrow \pi \oplus \langle a, k \rangle$ 
  until  $s \in G_\tau$ 
  return  $\pi$ 

function REFINE( $s, a$ )
  for  $b \leftarrow 1$  to  $B$  do
     $\mathcal{K} \leftarrow \text{SAMPLEPARAMS}(s, a, N)$ 
    if USEPRIOR then
       $\mathcal{K} \leftarrow \text{SORTASCENDING}(\mathcal{K}, V_{\text{A.P.}})$ 
    for all  $k \in \mathcal{K}$  do
       $s', e \leftarrow \text{SIMULATE}(s, a, k)$ 
      if  $e = \emptyset$  then
        return  $\langle s', k, \text{true} \rangle$ 
  return  $\langle s, \perp, \text{false} \rangle$ 

```

$\triangleright B$ batches of N
 $\triangleright$ same base sampler

Language (PDDL) [25], that 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 Planning Cost The anticipatory cost of a state s is the expected cost of solving future tasks from it [9–11],

$$V_{\text{A.P.}}^*(s) = \sum_{\tau'} P(\tau') V_{\tau'}^*(s), \quad (1)$$

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. Unlike prior work, which adds this cost to the planning objective, we leave the objective untouched and use the cost only to decide which sampled parameters to try first. Computing Eq. (1) exactly requires solving many tasks from every candidate state and is intractable during planning, so we learn a graph neural network (GNN) predictor $V_{\text{A.P.}}(s)$ over the scene graph (Sec. IV).

B. Anticipatory Planning Cost as a Sampling Prior

Algorithm 1 summarizes how the learned prior is integrated into the planner. The loop alternates between task planning and refinement, and the off-the-shelf solver is used as-is: the prior changes only how sampled candidates are ordered before they are checked (the **highlighted line**), while the base sampler, symbolic search, refinement, and replanning remain unchanged.

For each action a with continuous parameters, the base sampler proposes N candidates $\mathcal{K} = \{k_1, \dots, k_N\}$, each yielding a resulting state s_i . Instead of testing the candidates in draw order, we score each resulting state with the learned anticipatory cost $V_{\text{A.P.}}(s_i)$ and test candidates 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 fixed refinement budget; after that budget is exhausted, the failure

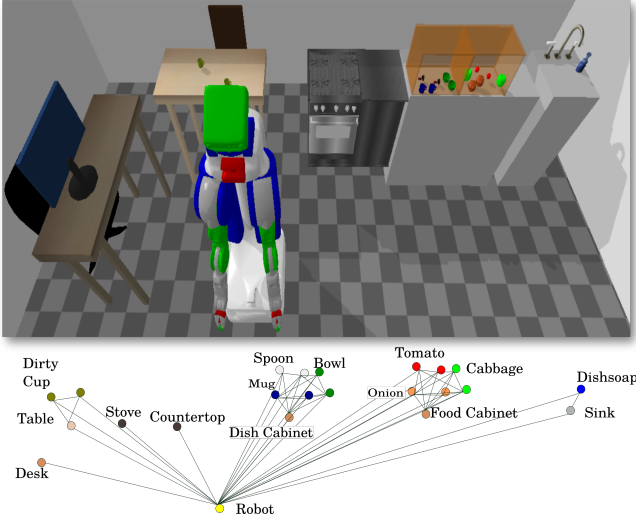


Fig. 2: Kitchen environment and its graph representation

is converted into symbolic facts Φ and the task is replanned (Alg. 1).

The uniform baseline follows the identical procedure with the prior disabled ($\text{USEPRIOR}=\text{false}$): each batch is drawn from the same base sampler and tested in draw order under the same budget. Thus sampler, candidate budget N , feasibility tests, and replanning logic are matched; the anticipatory prior only changes the within-batch ordering.

IV. ESTIMATING EXPECTED FUTURE COSTS VIA GRAPH NEURAL NETWORKS

Computing the expected future cost in Eq. (1) for every sampled candidate would require repeatedly solving many possible future tasks. We instead learn a predictor that maps the current environment state directly to $V_{\text{A.P.}}(s)$ and can be queried efficiently during refinement.

Graph Representation Each environment state is represented as a graph, where nodes correspond to entities in the scene and edges encode geometric relationships between them (Fig. 2). The node set contains one robot node, one node for each movable object, and region nodes corresponding to relevant workspace locations. Node features include entity-type encodings together with metric information derived from the current scene configuration, including object and robot poses. Edges are constructed between related entities and carry the inter-entity distance and an *obstacle count*, the number of objects lying between the two nodes.

Data Generation Training pairs $(s_i, V_{\text{A.P.}}(s_i))$ are generated offline (Fig. 1, top) by sampling environment states, drawing future tasks from the task distribution $P(\tau')$, solving each task from s_i with the TAMP solver, and aggregating the resulting costs via Eq. (1). Each state is converted into its graph and paired with its label. The resulting dataset contains 3000 training graphs and 1500 test graphs.

GNN Estimator We implement the anticipatory cost predictor using PyTorch Geometric [26, 27] with five TransformerConv message-passing layers [28], followed by global pooling and a linear regression head that maps the graph-

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.

N	Plan Time (s)		Search Evaluations	
	<u>Uniform</u>	<u>AntPrior</u>	<u>Uniform</u>	<u>AntPrior</u>
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

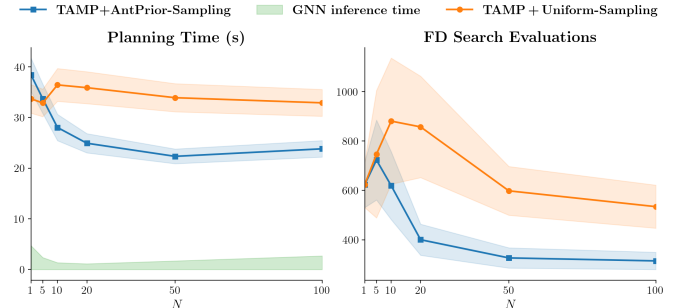


Fig. 3: Effect of the sampling budget. Both planning time and search evaluations with TAMP+ANTPRIOR are reduced compared to TAMP+UNIFORM. The green band shows the GNN inference time included in the planning time of TAMP+ANTPRIOR.

level embedding to a scalar estimate of $V_{\text{A.P.}}(s)$. The network is trained for 20 epochs using a learning rate of 0.05.

V. EXPERIMENTS AND RESULTS

We evaluate in a simulated kitchen implemented in Py-Bullet [29] and inspired by PIGINet [8]. The environment (Fig. 2) consists of a dining table, a long counter with multiple cabinet bins, a sink, and a stove. Tasks move objects between locations and require reasoning over obstruction, clutter, large combinatorial search spaces, and geometric feasibility. The domain has three operators: *move* (with cost based on Euclidean distance), and *pick* and *place* (each with a fixed cost of 20 units).

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 the N candidates in draw order.

TAMP+ANTPRIOR tries the N candidates in increasing $V_{\text{A.P.}}$ order.

We vary the candidate budget $N \in \{1, 5, 10, 20, 50, 100\}$ and report planning time, which for TAMP+ANTPRIOR includes GNN inference, and the number of Fast Downward search evaluations as a measure of symbolic search effort.

At a matched sampling budget, the prior reaches the goal with fewer search evaluations and less planning time than uniform sampling (Table I, Fig. 3), improving symbolic-search efficiency by up to 53% (at $N=20$) and planning time by up to 34% (at $N=50$). For every $N \geq 10$,

TAMP+ANTPRIOR reduces planning time by 23–34% and search evaluations by 30–53%.

At $N=1$ there is a single candidate to rank, so both methods incur identical search evaluations (623); the residual time gap is the GNN querying time itself. At $N=5$ the pool is too small for ranking to matter, and the two methods are within noise of each other. From $N=10$ onward the prior pulls clearly ahead, and 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, so planning time rises slightly at $N=100$ even as search evaluations continue to fall.

VI. CONCLUSION

We demonstrate that a learned anticipatory planning cost, originally developed to reason about the expected cost of future tasks can also serve as 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*, vol. 4, pp. 265–293, 2021.
- [2] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical task and motion planning in the now,” in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1470–1477.
- [3] E. Plaku and G. D. Hager, “Sampling-based motion and symbolic action planning with geometric and differential constraints,” in *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 5002–5008.
- [4] M. Toussaint, “Logic-geometric programming: An optimization-based approach to combined task and motion planning,” in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.
- [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 *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 639–646.
- [6] 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)*. PMLR, 2021, pp. 64–79.
- [7] 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 *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 447–454. [Online]. Available: <https://people.eecs.berkeley.edu/~pabbeel/papers/2016-ICRA-tamp-learning.pdf>
- [8] 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)*, Daegu, Republic of Korea, July 2023.
- [9] R. Dhakal, M. R. H. Talukder, and G. J. Stein, “Anticipatory planning: Improving long-lived planning by estimating expected cost of future tasks,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 538–11 545.
- [10] 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 *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [11] 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*, vol. 11, no. 2, pp. 1850–1857, 2026.
- [12] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning,” in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 30, 2020, pp. 440–448.
- [13] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, “Learning to guide task and motion planning using score-space representation,” *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 793–812, 2019.
- [14] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4107–4114.
- [15] R. Chitnis, L. P. Kaelbling, and T. Lozano-Pérez, “Learning quickly to plan quickly using modular meta-learning,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7865–7871.
- [16] J. Mendez-Mendez, L. P. Kaelbling, and T. Lozano-Pérez, “Embodied lifelong learning for task and motion planning,” in *Conference on Robot Learning (CoRL)*. PMLR, 2023.
- [17] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, “Learning feasibility for task and motion planning in tabletop environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1255–1262, 2019.
- [18] S. Ait Bouhsain, R. Alami, and T. Siméon, “Learning to predict action feasibility for task and motion planning in 3D environments,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.
- [19] B. Kim and L. Shimanuki, “Learning value functions with relational state representations for guiding task-and-motion planning,” in *Conference on Robot Learning (CoRL)*. PMLR, 2020, pp. 955–968.
- [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 *Robotics: Science and Systems*. Corvallis, OR: RSS Foundation, 2020.
- [21] Q. Du, B. Li, Y. Du, S. Su, T. Fu, Z. Zhan, Z. Zhao, and C. Wang, “Fast task planning with neuro-symbolic relaxation,” *IEEE Robotics and Automation Letters*, 2026. [Online]. Available: <https://arxiv.org/abs/2507.15975>
- [22] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, “Learning symbolic operators for task and motion planning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3182–3189.
- [23] R. Arora, S. Singh, K. Swaminathan, A. Datta, S. Banerjee, B. Bhowmick, K. M. Jatavallabhula, M. Sridharan, and M. Krishna, “Anticipate & Act: Integrating LLMs and classical planning for efficient task execution in household environments,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
- [24] M. Helmert, “The Fast Downward planning system,” *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006. [Online]. Available: <https://ai.dmi.unibas.ch/papers/helmert-jair06.pdf>
- [25] M. Fox and D. Long, “PDDL2.1: An extension to PDDL for expressing temporal planning domains,” *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.
- [26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems* 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.
- [27] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [28] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, “Masked label prediction: Unified message passing model for semi-supervised classification,” 2021.
- [29] E. Coumans and Y. Bai, “PyBullet, a Python module for physics simulation for games, robotics and machine learning,” 2016–2021.