TGPO: Temporal Grounded Policy Optimization for Signal Temporal Logic Tasks

Yue Meng **Fei Chen** **Chuchu Fan**
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
{mengyue,feic,chuchu}@mit.edu

Abstract

Learning control policies for complex, long-horizon tasks is a central challenge in autonomous systems. Signal Temporal Logic (STL) offers an expressive language for specifying such tasks, but its non-Markovian nature and inherent sparse reward make it hard for standard Reinforcement Learning (RL) to solve. Prior RL approaches focus only on limited STL fragments or use robustness scores as sparse rewards. Our new method TGPO (Temporal Grounded Policy Optimization) decomposes STL into timed subgoals and invariant constraints and tackles the problem in a hierarchical fashion. Its high-level module proposes time allocations for subgoals, and the low-level time-conditioned policy learns to achieve the sequenced subgoals using a dense, stage-wise reward. In inference, we use the critic to efficiently sample time allocations and select the most promising assignment for the policy to rollout. We evaluate in five environments, from low-dimensional navigation to manipulation, drone, and quadrupedal locomotion, and TGPO significantly outperforms baselines (especially for high-dimensional and long-horizon cases), with 31.6% higher success rate compared to the best method.

1 Introduction

Signal Temporal Logic (STL) is a powerful framework for specifying tasks with temporal and spatial constraints in real-world robotic applications. However, designing controllers to satisfy these specifications is difficult, especially for systems with complex dynamics and a long task horizon. While Reinforcement Learning (RL) excels in handling these dynamical systems, directly deploying RL for STL specifications poses significant challenges. The history-dependent nature of STL breaks the Markovian assumption for the common RL algorithms. Furthermore, the reward based on the STL satisfaction is extremely sparse for long-horizon tasks, making RL struggle to learn effectively.

Existing model-free RL approaches for STL tasks typically leverage state augmentation with reward shaping. τ-MDP [1] encodes histories explicitly in the augmented spaces and F-MDP [50] designs flags to bookkeep the satisfaction of STL subformulas. However, these techniques only work on limited STL fragments with up to two temporal layers. While model-based RL [20, 16] has fewer restrictions on the STL formulas, learning the system (latent space) dynamics can be challenging, and the estimation error accumulates over long horizons. Additionally, the planning often relies on Monte Carlo Tree Search or sampling action sequences, which is intractable for high-dimensional systems.

We argue that the primary barrier for RL to efficiently solve STL tasks is the difficulty of designing a dense, stage-wise reward function. This challenge stems directly from the unspecified temporal variables governing the "reach"-type tasks in STL formulas, which prevents a direct decomposition of STL into a sequence of executable subgoals. For example, for an STL $F_{[0,160]}A \wedge F_{[0,160]}B$ ("Eventually reach A and eventually reach B within the time interval $[0, 160]$"), the time assignments

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

for reaching A and reaching B determine the order of visiting these regions. If we can ground the variables into concrete values (e.g., reach A at 35, and reach B at 120), the problem can be cast into sequential goal-reaching, which is much easier to solve.

Inspired by this observation, we propose a hierarchical RL framework to solve STL tasks by iteratively conducting **T**emporal **G**rounding and **P**olicy **O**ptimization (TGPO). The high-level component assigns values for the time variables to form the sequenced subgoals, and the low-level time-conditioned policy learns to achieve the task guided by the dense, stage-wise rewards derived from these subgoals. To efficiently bind values for multiple time variables, we carry out a high-level temporal search with a critic that predicts STL satisfaction. A Metropolis–Hastings sampling is used to guide exploration toward more "promising" time allocations. During inference, we sample time variable assignments and evaluate them using the critic. The most promising schedule is then executed by the low-level policy to generate the final solution for the STL specification.

We conduct extensive experiments over five simulation environments, ranging from 2D linear dynamics to 29D Ant navigation tasks. Compared to other baselines, TGPO* (with Bayesian time variable sampling) achieves the highest overall task success rate. The performance gains are significant, especially in high-dimensional systems and long-horizon tasks. Furthermore, our time-conditioned design offers key benefits: our critic offers interpretability by identifying promising temporal plans, and the policy can generate diverse, multi-modal behaviors to satisfy a single STL specification.

Our contributions are as follows: (1) **Hierarchical RL-STL framework**: To the best of our knowledge, we are the first to develop a hierarchical model-free RL algorithm capable of solving deeply-nested, finite-time STL tasks over long horizons. (2) **Critic-guided Bayesian sampling**: We introduce a critic-guided temporal grounding mechanism that, together with STL decomposition, yields subgoals and invariant constraints. This mechanism constructs an augmented MDP with dense, stage-wise rewards and thus overcomes the sparse reward challenges that have hindered existing RL approaches. (3) **Interpretability**: By explicitly grounding subgoals and invariant constraints in the STL structure using critic-guided Bayesian sampling, our approach offers a more interpretable learning process, where progress can be directly traced to logical task components. (4) **Complex dynamics and reproducibility**: TGPO demonstrates strong performance over other baselines and fits for complex dynamics, which supports the effectiveness of the design. All code, including the proposed algorithm, simulations, and STL tasks, is publicly available at: `https://github.com/mengyuest/TGPO`.

2 Related Work

2.1 Signal Temporal Logic tasks

Signal Temporal Logic (STL) offers a powerful framework for specifying robotics tasks [8]. Compared with Linear Temporal Logic (LTL), STL operates over continuous signals with explicit timing constraints and real-valued predicates, making it challenging to conduct planning [11]. Traditional approaches for STL include sampling-based methods [49, 22, 28, 42], mixed-integer programming [45, 23, 21], reachability-based control synthesis [13, 52], and trajectory optimization [24]. More recently, learning-based methods have emerged, including differentiable policy learning [30, 31, 34], imitation learning [38, 25, 35, 36], and RL [27].

2.2 Reinforcement learning for temporal logic tasks

Temporal logic RL has been extensively studied in LTL and some STL fragments [27], where the key challenge is designing suitable rewards. For LTL, existing methods [39, 26, 14, 15] typically convert the formula into Limit-Deterministic Büchi Automata (LDBA) [43] or reward machines [17], while LTL2Action [47] uses progression [2] to assign dense reward, and SpectRL [19] devises a composable specification language for complex objectives. In contrast, STL poses additional challenges due to its explicit time constraints and real-value predicates. Early approaches augment the state space via temporal abstractions using history segments [1, 18] or flags [50, 51], while bounded horizon nominal robustness (BHNR) [3] offers intermediate reward approximations. Recent work uses model-based learning to solve STL tasks with evolutionary strategies [20] and Monte Carlo Tree Search in value function space [16]. However, most of these methods are restricted to STL structures and systems (limited temporal nesting, fixed-size time windows, or grid-like environments). Instead, our method can handle complex STLs with efficiently designed augmented states and dense stage-wise rewards.

3 Preliminaries

3.1 Signal Temporal Logic (STL)

Consider a discrete-time system $x_{t+1} = f(x_t, u_t)$ where $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$ denote the state and control at time t. Starting from an initial state x_0, a signal $\sigma = x_0, ..., x_T$ is generated via controls $u_0, ..., u_{T-1}$. STL specifies properties via following rules [9]:

$$\phi ::= \top \mid \mu(x) \geq 0 \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U_{[a,b]} \phi_2. \tag{1}$$

Here the boolean-type operators split by "|" are the building blocks to compose an STL: $\top$ means "true", μ denotes a predicate function $\mathbb{R}^n \to \mathbb{R}$, and $\neg$, $\wedge$, U, $_{[a,b]}$ are "negation", "conjunction", "until" and the time interval from a to b. Other operators are "disjunction": $\phi_1 \vee \phi_2 = \neg(\neg\phi_1 \wedge \neg\phi_2)$, "eventually": $F_{[a,b]}\phi = \top U_{[a,b]}\phi$ and "always": $G_{[a,b]}\phi = \neg F_{[a,b]}\neg\phi$. We denote $\sigma, t \models \phi$ if the signal σ from time t satisfies the STL formula (the evaluation of ϕ returns True). In particular, we simply write $\sigma \models \phi$ if the signal is evaluated from $t = 0$. For operators $\top, \mu \geq 0, \neg, \wedge$ and $\vee$, the evaluation checks for the signal state at time t. As for temporal operators [32], the semantics are defined as follows: $\sigma, t \models F_{[a,b]}\phi \Leftrightarrow \exists t' \in [t+a, t+b]$, $\sigma, t' \models \phi$; $\sigma, t \models G_{[a,b]}\phi \Leftrightarrow \forall t' \in [t+a, t+b]$, $\sigma, t' \models \phi$; and $\sigma, t \models \phi_1 U_{[a,b]}\phi_2 \Leftrightarrow \exists t' \in [t+a, t+b], \sigma, t' \models \phi_2, \forall t'' \in [t, t'], \sigma, t'' \models \phi_1$. In plain words, $\phi_1 U_{[a,b]}\phi_2$ means "ϕ_1 holds until ϕ_2 happens in $[a, b]$." Robustness score [10] $\rho(\sigma, t, \phi)$ measures how well a signal σ satisfies ϕ. We have $\rho \geq 0$ iff $\sigma, t \models \phi$. The score ρ is:

$$\begin{aligned}
&\rho(\sigma, t, \top) = 1 \\
&\rho(\sigma, t, \mu) = \mu(\sigma(t)) \\
&\rho(\sigma, t, \neg\phi) = -\rho(\sigma, t, \phi) \\
&\rho(\sigma, t, \phi_1 \wedge \phi_2) = \min\left(\rho(\sigma, t, \phi_1), \rho(\sigma, t, \phi_2)\right) \\
&\rho(\sigma, t, F_{[a,b]}\phi) = \sup_{r \in [a,b]} \rho(\sigma, t+r, \phi) \\
&\rho(\sigma, t, G_{[a,b]}\phi) = \inf_{r \in [a,b]} \rho(\sigma, t+r, \phi) \\
&\rho(\sigma, t, \phi_1 U_{[a,b]}\phi_2) = \sup_{t' \in [t+a, t+b]} \min\left\{\rho(\sigma, t', \phi_2), \inf_{t'' \in [t, t']} \rho(\sigma, t'', \phi_1)\right\}
\end{aligned} \tag{2}$$

3.2 Markov Decision Process

A Markov Decision Process (MDP) is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where: $\mathcal{S}$ and $\mathcal{A}$ represent the sets of states and actions, respectively, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the probabilistic transition function where $P(s'|s, a)$ denotes the probability of the next state s' given current state s and action a, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The agent decision is made by a policy $\pi : \mathcal{S} \to \mathcal{A}$ which maps states to a probability distribution over actions. The objective is to find an optimal policy π^* that maximizes the expected discounted cumulative reward from a starting state s_0: $\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0\right]$ with $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim P(\cdot|s_t, a_t)$.

3.3 Problem formulation

Consider a discrete-time system with state space $\mathcal{X}$, control space $\mathcal{U}$ and the initial state distribution ρ_0 over the support $\mathcal{X}_0 \subseteq \mathcal{X}$. Given an STL ϕ defined in Eq. 1, our objective is to learn a policy π that can generate a trajectory σ to maximize STL satisfaction probability: $\max_\pi \mathbb{P}_{x_0 \sim \rho_0}(\sigma \models \phi)$.

Remarks. It is tempting to directly solve it via MDP, treating the control system state $\mathcal{X}$ as the MDP state $\mathcal{S}$, and the control input $\mathcal{U}$ as the action space $\mathcal{A}$. However, for STL tasks, the policy also depends on the history[1], making the problem non-Markovian. Thus, we need to augment the MDP state to keep history data. Besides, the satisfaction of an STL is checked over the full trajectory, making it difficult to define dense rewards (unlike LTL, where stage-wise rewards [5, 47] can be defined). Thus, we need to design dense rewards under the augmented state space to learn efficiently.

[1] E.g., if an STL task is to "Eventually reach region A and then reach B", the policy needs to "remember" whether it has already visited the region A in order to proceed to reach B.

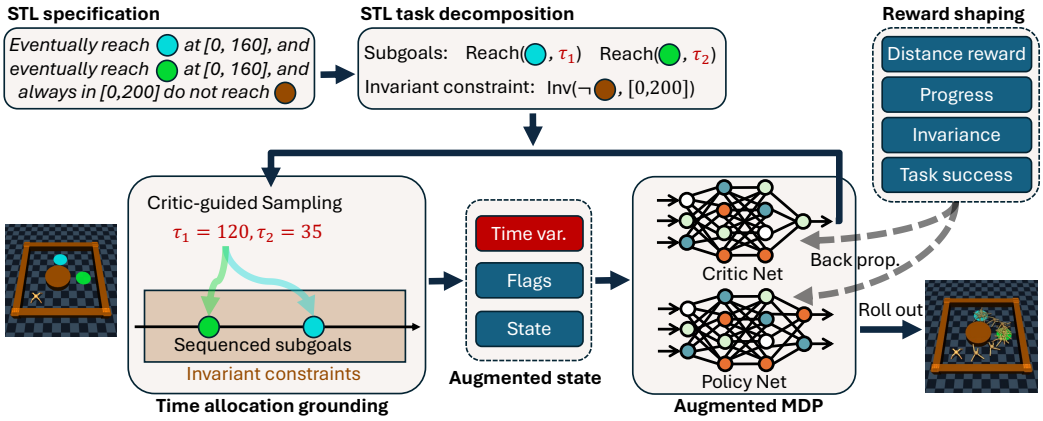


Figure 1: Framework: STL decomposition and critic-guided temporal grounding yield subgoals and invariant constraints that guide an augmented MDP with dense rewards for policy optimization.

4 Methodology

We propose TGPO, Temporal Grounded Policy Optimization, to address the problem above. The framework is illustrated in Fig. 1, and we will explain it in detail below.

4.1 STL subgoal decomposition

The inherent tree structure of STL facilitates its decomposition into subgoals with invariant constraints [21, 29, 52]. Our framework leverages this property to map a global specification ϕ onto a set of subtasks. Each subtask is evaluated via a predicate function μ over the trajectory trace σ and is classified into one of the following types:

- **Reachability task**: achieve $\mu(\sigma(\tau)) \geq 0$ at a time instant τ, denoted as $\text{Reach}(\mu, \tau)$.
- **Invariance task**: keep $\mu(\sigma(\tau)) \geq 0$ for all time τ in an interval W, denoted as $\text{Inv}(\mu, W)$.

For basic STL formulas, the time instants and the time intervals can be concrete values or variables: e.g., the formula $G_{[a,b]}\mu$ can be written as $\text{Inv}(\mu, [a, b])$ with concrete $[a, b]$, whereas the formula $F_{[a,b]}\mu$ can be written as $\text{Reach}(\mu, \tau)$ with the time variable $\tau \in [a, b]$, and $\mu_1 U_{[a,b]}\mu_2$ can be written as $\{\text{Reach}(\mu_2, \tau), \text{Inv}(\mu_1, [a, \tau])\}$ with the time variable $\tau \in [a, b]$. For a nested STL, we follow a top-down approach to "flatten" it into reachability and invariance tasks governed by time variables. We denote $\text{Reach}(\phi, \tau)$ for $\rho(\sigma, \tau, \phi) \geq 0$ and use $\text{Inv}(\phi, W)$ to represent $\rho(\sigma, \tau, \phi) \geq 0, \forall \tau \in W$. For any STL ϕ we can write it as $\text{Reach}(\phi, 0)$ and then we rewrite with tasks using its subformulas. The subformula will always carry time variables from its ancestor operators, and we repeat the process until all the tasks are represented as atomic propositions (APs) corresponding to μ or its negation $\neg\mu$. For example, for $\phi = F_{[a,b]}\phi_0 \wedge G_{[c,d]}\neg\mu_0$ where $\phi_0 = \mu_1 \wedge G_{[a_2,b_2]}\mu_2 \wedge F_{[a_3,b_3]}\mu_3$ is a subformula, we can represent ϕ as $\{\text{Reach}(\phi_0, \tau), \text{Inv}(\neg\mu_0, [c, d])\}$ with domain $\{\tau \in [a, b]\}$, then we can pass τ into ϕ_0 to represent the STL as $\{\text{Reach}(\mu_1, \tau), \text{Inv}(\mu_2, [\tau+a_2, \tau+b_2]), \text{Reach}(\mu_3, \tau+\tau'), \text{Inv}(\neg\mu_0, [c, d])\}$ with domains $\{\tau \in [a, b], \tau' \in [a_3, b_3]\}$. An illustration of the decomposition is depicted in Fig. 2. In this work, disjunctions or temporal structures of the form "$G(F \ldots)$" can be addressed by introducing extra binary variables to determine the active disjunction branch (shown in App. A.11) and more time variables for each instant in G operator temporal domain.

From the reachability and invariance tasks, we further denote **subgoals** (reach or stay) as tasks that are either a reachability task (e.g., **Subgoal 1** in Fig. 2) or an invariance task (e.g., **Subgoal 2** in Fig. 2) with atomic proposition μ (we assume all the APs are for reaching certain regions). The remaining invariance tasks associated with negation of APs (e.g., $\text{Inv}(\neg\mu_0, [c, d])$) are treated as **invariant constraints** (avoidance). Through this decomposition, a complex STL formula ϕ reduces to N_g subgoals ϕ_i^g with $\text{Reach}(\mu_i^g, \tau_i)$ or $\text{Inv}(\mu_i^g, W_i)$, $i \in \{1, \cdots, N_g\}$ and N_c invariant constraints ϕ_j^c with $\text{Inv}(\neg\mu_j^c, W_j)$, $j \in \{1, \cdots, N_c\}$. Denote Θ_g and Θ_c as the sets of subgoals and invariant constraints, respectively. Each subgoal / constraint has a starting time and an ending time $[\underline{t}, \bar{t}]$ which

is $[\tau, \tau]$ (or W). Denote all the time variables in this STL as $\mathbf{t}$. The full algorithm for the temporal decomposition is shown as Algo. 2 in App. A.3. Next, we will show how this decomposition guides our state augmentation and reward shaping.

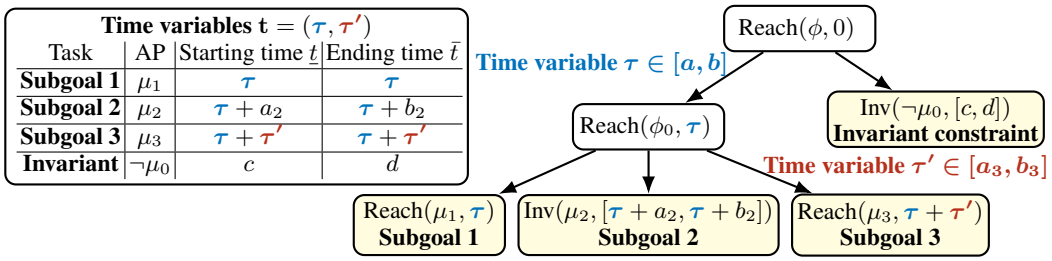


Figure 2: STL decomposition of $\phi = F_{[a,b]}(\mu_1 \wedge G_{[a_2,b_2]}\mu_2 \wedge F_{[a_3,b_3]}\mu_3) \wedge G_{[c,d]}\neg\mu_0$.

4.2 Temporal grounded state augmentation and reward design

Given a concrete time variables assignment $\mathbf{t}$, the problem is now structured as reaching a sequence of subgoals sorted by their starting time with invariant constraints satisfied during execution. For brevity, we assume the subgoal indices are already sorted. We augment our state as:

$$s = (x, \tau, p_{prev}, p, r, \chi) \tag{3}$$

Here $x \in \mathbb{R}^n$ stands for the original state from the discrete-time system mentioned in Sec. 3.3, $\tau \in \{0, 1, \cdots, T\}$ represents the time index, $p \in \{0, 1, \cdots, N_g\}$ represents the progress index and p_{prev} records the previous progress, r records the certificate to proceed to the next subgoal, $\chi \in \{0, 1\}^{N_c}$ maintains the satisfaction status for the invariant constraints. For the k-th subgoal (or invariant constraint), denote the starting time $\underline{t}^g_k$ (or $\underline{t}^c_k$) and the ending time $\bar{t}^g_k$ (or $\bar{t}^c_k$). The augmented state transition can be written as:

$$\begin{cases} x' = f(x, u), \quad \tau' = \tau + 1, \quad p'_{prev} = p, \quad r' = h(r, x', \tau', p'), \quad p' = p + \mathbb{1}(r = 2), \\ \chi'_k = \chi_k \times \mathbb{1}(\neg(\underline{t}^c_k \leq \tau' \leq \bar{t}^c_k \wedge \neg\mu^c_k(x') < 0)) \qquad k = 0, 1, ..., N_c \end{cases} \tag{4}$$

where $h(r, x', \tau', p')$ is calculated as:

$$\begin{cases} 0, & \text{if } r = 2 \\ 1, & \text{if } \underline{t}^g_{p'} \neq \bar{t}^g_{p'} \wedge \tau' = \underline{t}^g_{p'} \wedge \mu^g_{p'}(x') \geq 0 \\ 2, & \text{if } (r = 1 \vee \underline{t}^g_{p'} = \bar{t}^g_{p'}) \wedge (\tau' = \bar{t}^g_{p'} \wedge \mu^g_{p'}(x') \geq 0) \\ r, & \text{otherwise} \end{cases} \tag{5}$$

The variable r acts as a certificate (or flag) that keeps track of whether the reach-and-stay (FG) condition has been satisfied. It encodes the progress toward establishing that the predicate holds both at the entry time and the exit time of the required interval. To guide the agent to achieve these subgoals in a proper time window while satisfying the invariant constraints, we design the reward in the following form ($\lambda_1 > 0, \lambda_2 > 0, \lambda_3 > 0, \lambda_4 < 0, \lambda_5 > 0$):

$$R(s) = \lambda_1 R_{dist} + \lambda_2 R_{progress} + \lambda_3 R_{success} + \lambda_4 R_{inv} + \lambda_5 R_{robustness} \tag{6}$$

where $R_{dist} = \mu^g_p(x)$ is a distance-based reward shaping to encourage the agent to reach the current subgoal (and stay at the current subgoal within the time window $[\underline{t}^g_p, \bar{t}^g_p]$), $R_{progress} = \mathbb{1}(p^{prev} \neq p)$ encourages the agent to achieve more subgoals, $R_{success} = \mathbb{1}(p = N_g \wedge \chi = \mathbf{1})$ encourages the agent to finish all subgoals without violating any invariant constraints, and $R_{inv} = \mathbb{1}(\chi_k = 0)$ penalizes for violating invariant constraints. The robustness score $R_{robustness}$ is also used at the final time step to encourage the agent to satisfy the STL specification. In this way, the agent is incentivized to reach all the subgoals while obeying the invariant constraints (the proof of reward consistency towards STL satisfaction is shown in App. A.2). We use a consistent set of values for reward weights $\lambda_{\{1,2,3,5\}}$ across all environments and only tune the invariance weight λ_4 for different environments. The weights are fixed across different tasks; we always start with a simple task to tune the reward weights and then keep them fixed for the remaining tasks. We use Proximal Policy Optimization

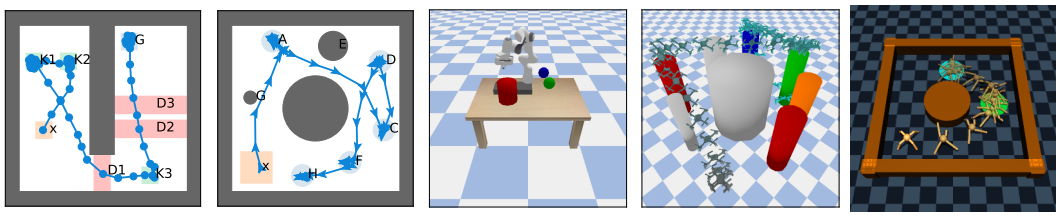
(PPO) [41] to train the agent. The policy network and the critic receive the augmented state and the time variable assignment as the input, and output the action and the critic value correspondingly. In the implementation, the time index feature input τ is normalized between 0 and 1. At the beginning of each training epoch, we sample the time variables and collect episodes to update the network parameters. During inference, we sample time variables and use the trained critic to find the most effective assignment. A naive way to sample time variables is randomly sampling from their feasible intervals, which is not data efficient. We will present a better solution in the next section.

Algorithm 1 TGPO with Hybrid Time Variable Sampling

Input: STL formula ϕ (subgoals and invariant constraints), elite buffer size K, batch size N_B
Initialize policy $\pi_\theta(a|s, \mathbf{t})$, critic $V_\psi(s, \mathbf{t})$, and elite time variable buffer $\mathcal{B}$
for iteration $i = 1, \ldots, N$ **do**
  // 1. High-level Temporal Grounding
  $\mathbf{T}_{\text{uniform}} \leftarrow$ Sample $\eta_{\text{uniform}} N_B$ time variables uniformly from the valid domain $\mathcal{T}$
  $\mathbf{T}_{\text{mcmc}} \leftarrow$ Run Metropolis-Hastings guided by V_ψ to generate $\eta_{mcmc} N_B$ time variables.
  $\mathbf{T}_{\text{elite}} \leftarrow$ Top $\eta_{elite} N_B$ time variables from elite buffer $\mathcal{B}$
  $\mathbf{T}_{\text{batch}} \leftarrow \mathbf{T}_{\text{uniform}} \cup \mathbf{T}_{\text{mcmc}} \cup \mathbf{T}_{\text{elite}}$
  // 2. Low-level Policy Optimization
  Collect trajectories $\mathcal{D}_i = \{(\sigma_j, \rho_j^\phi, \mathbf{t}_j)\}$ by executing π_θ with time variables from $\mathbf{T}_{\text{batch}}$
  Update π_θ and V_ψ using the PPO algorithm on $\mathcal{D}_i$
  Update $\mathcal{B}$ with time variables from $\mathcal{D}_i \cup \mathcal{B}$ corresponding to top-K STL robustness score
end for
return Trained policy π_θ, critic V_ψ, and elite buffer $\mathcal{B}$.

4.3 Critic-guided Bayesian time allocation

The key challenge in our framework is efficiently searching for time variable assignments. A naive uniform sampling strategy might waste huge effort on assignments that lead to infeasible or low-reward trajectories. To address this, we propose a Bayesian sampling strategy to find promising time assignments. We do not need to learn an extra surrogate function, as the value function already provides a powerful heuristic. Our prior belief towards "promising" time variables is the uniform distribution. At each epoch, we update our belief using the learned critic value as a non-normalized approximation of the log-likelihood of success, and then we sample the time variables using the Metropolis-Hastings (MH) algorithm. The MH performs a guided random walk over the discrete time variable space and prefers to stay in regions that yield high critic values. To mitigate the risk of the sampler converging to a local optima and the fact that the initial critic might not be accurate, we adopt a hybrid approach: In each epoch, we use an MH sampler to obtain a ratio η_{mcmc} of the time variables and sample a ratio η_{uniform} through uniform sampling. To further leverage knowledge across training epochs, we maintain a replay buffer containing the top η_{elite} ratio of "elite" time variable assignments that yield the highest STL robustness scores. This combination creates a robust and efficient mechanism for discovering effective temporal plans. The full training procedure is detailed in Algo. 1, the MH algorithm is in Algo. 3 (App. A.7), and the ablation study is shown in Sec. 5.4.



(a) Linear (b) Unicycle (c) Franka Panda (d) Quadrotor (e) Ant

Figure 3: Simulation benchmarks.

5 Experiments

5.1 Implementation details

Baselines. We consider the following approaches. **RNN**: Train RL with a recurrent neural network (RNN) to handle history and use the STL robustness score as the rewards. **CEM**: Cross-Entropy Method [7] that optimizes the policy network with the robustness score as the fitness score. **Grad**: A gradient-based method [34] that trains the policy with a differentiable STL robustness score. **τ-MDP**: An RL method [1] which augments the state space with a trajectory segment to handle history data. **F-MDP**: An RL approach [50] that augments the state space with flags. We denote our base method as **TGPO** and the enhanced version with Bayesian sampling as **TGPO***. Details are in App. A.6.

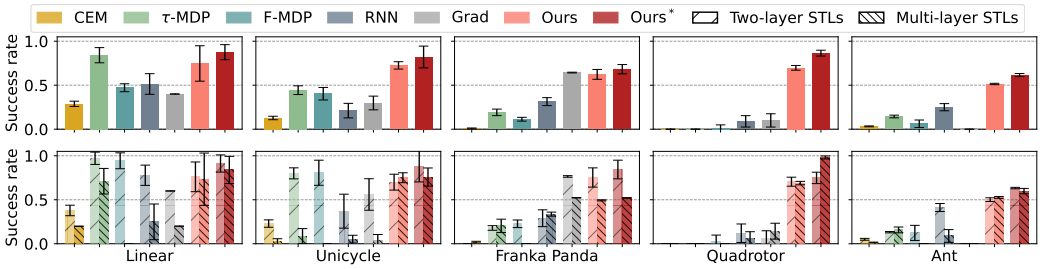


Figure 4: Main comparison with learning-based approaches. Our method has a higher task success rate compared to other baselines. The error bar shows the standard deviation of success rate over seven random seeds. In the appendix, we show comparisons with classical approaches (App. A.9) and aggregation-based approaches (App. A.10).

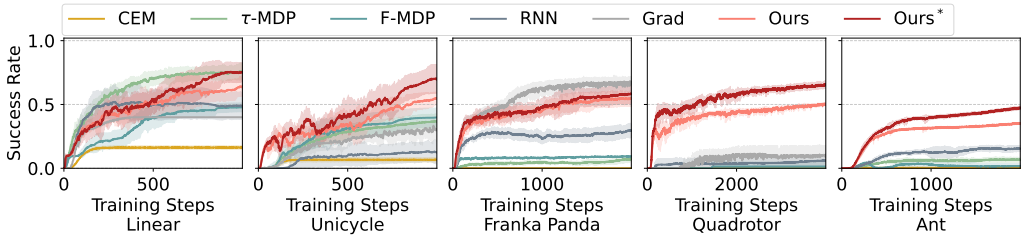


Figure 5: STL success rate evaluation (under seven random seeds) along the training process.

Benchmarks. We evaluate TGPO across five environments shown in Fig. 3 with varying dynamics and dimensionality: (1) **Linear**: A 2D point-mass linear system. (2) **Unicycle**: A non-holonomic 4D system for a wheeled robot. (3) **Franka Panda**: A 7-DoF robot arm. (4) **Quadrotor**: A 12D, full dynamic model of a quadrotor. (5) **Ant**: a 29D quadruped robot for locomotion tasks. We specify the regions that the agent needs to reach, stay, or avoid and specify their temporal constraints using STL specifications. We designed 10 STL tasks for each benchmark. Five of them are two-layered (e.g., $F_{[0,T]}G_{[0,5]}$(Reach A)), solvable by all the methods. The rest are multi-layer STLs with single-layer or deeper nesting, which cannot be solved by F-MDP. Details can be found in App. A.16.

Training and evaluation. For the main comparisons, the task horizon is fixed at T=100 except for "Ant" (T=200). We trained each model with 7 random seeds to ensure statistical significance. All the methods are implemented in JAX [4] and trained with 512 parallel environments for 1000$\sim$4000 epochs. All experiments were conducted on Amazon Web Services (AWS) g6e.2xlarge instances (each one has an NVIDIA L40S GPU). A single experiment (a specific set of environment, method, STL, and random seed) took 5 to 90 minutes, depending on the environment and method complexity. In the testing stage, we sample 512 initial states. For each initial state, each baseline is given 10 attempts to generate the solution, and the trajectory with the highest STL robustness score is selected. For our approach, we attempt to select the best time assignment only once, based on the critic value, and then roll out the trajectory (we avoid the use of the STL score as feedback to choose the trajectory). The **Success rate** is the average performance over all the initial states and the STLs. We also measured **Training time** for the additional experiments in App. A.8 and App. A.9.

5.2 Main results

As shown in Fig. 4 (top row), TGPO achieves the leading performance in most benchmarks, and with Bayesian time variable sampling, TGPO* achieves the highest overall success rate across all benchmarks, indicating the strong empirical performance. Our advantage becomes clearer as the system dimension and the planning difficulty increase, especially in "Quadrotor" and "Ant", where most of the baselines achieve less than 10% success rate, whereas TGPO* can achieve 86.46% and 61.57% success rate, respectively. Under "Linear" system, the best baseline τ-MDP (84.11%) performs competitively compared to TGPO* (87.53%), but τ-MDP's performance drops drastically on the other benchmarks. The "Grad" method is a strong baseline on "Franka Panda", however, its success rate decreases by a large margin on "Quadrotor" due to its complex nonlinear dynamics, and it cannot work at all on "Ant", which is likely caused by the discrepancy between the simulator's approximated gradients and the true non-differentiable dynamics. These findings showcase TGPO's strong performance and great adaptation to high-dimensional and non-differentiable environments. If we look at different types of STLs (Fig. 4, bottom row), on low-dimensional cases ("Linear" and "Unicycle"), most baselines work well under the simple STL tasks ("two-layer STLs") but they struggle on the harder STLs ("multi-layer STLs", note that F-MDP can only handle "two-layer STLs"). Whereas TGPO and TGPO* excel at working on these complex STLs and perform consistently well. This shows our approach's strength in handling complex STLs. Fig. 5 shows the task success rate in training. Our approach can achieve a high task success rate eventually, whereas other baselines plateau early in the training. Additional comparison with traditional methods is in App. A.9. Additional comparison with aggregation-based methods is in App. A.10.

5.3 Solving STL specifications with different horizon lengths

Beyond system complexity and task difficulty, our methods also show resilient adaptivity for long-horizon tasks. Here, we consider only the two-layer STLs and we scale the task horizon to different lengths (50, 200, 300, 800 and 1000). As shown in Fig. 6, our methods (TGPO and TGPO*) keep a high success rate over varied time lengths, whereas for RL methods τ-MDP, F-MDP and RNN, which are strong baselines for shorter horizons (T=50 and 100), experience a huge drop in success rate as the horizon increases. It is interesting that CEM and Grad can maintain their performance as the horizon expands 10 times, which may be attributed to their trajectory optimization formulation.

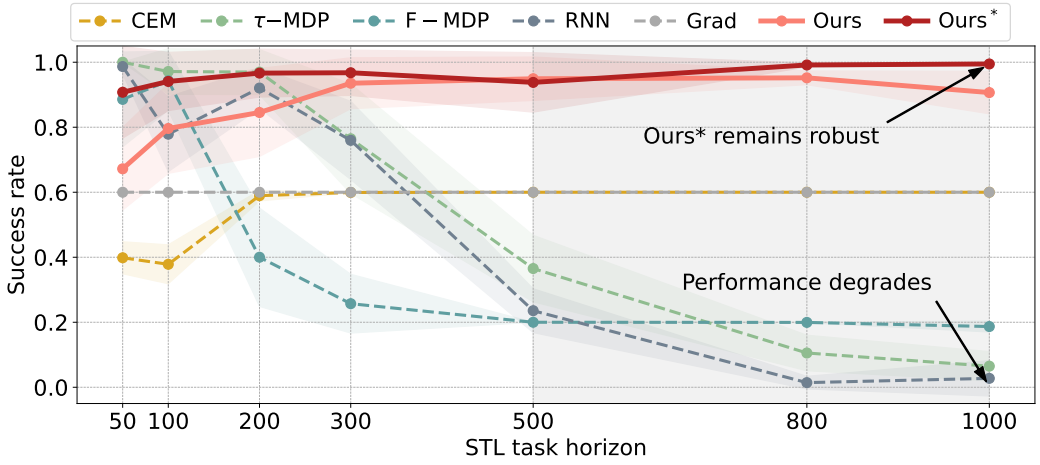


Figure 6: Solving STL in "Linear" environment over varied task horizons. Our method performs the best and maintains high success rate in long horizons where the RL baselines performance degrades.

5.4 Ablation studies

We conduct a thorough ablation study under "Linear" (all 10 STLs) for the analysis. We first study different sampling strategies. As shown in Tbl. 1a, our base model with random sampling (Ours) can already achieve 80.33% success rate ($\pm$ indicates the standard deviation over 7 random seeds). However, naively using Bayesian sampling (Ours_{Bay}) or Elite variable replay buffer ($\text{Ours}_{\text{Elite}}$) will

Table 1: Ablation studies for TGPO on the linear dynamics environment.

(a) Different time variables sampling strategies.

Method	Rand.	Bay.	Elite	Test(%)
Ours	✓			80.33 ± 8.84
Ours_{Bay}		✓		53.79 ± 7.99
$\text{Ours}_{\text{Elite}}$			✓	61.49 ± 10.02
$\text{Ours}_{\text{mixBay}}$	✓	✓		81.18 ± 9.72
$\text{Ours}_{\text{mixElite}}$	✓		✓	86.62 ± 8.67
$\text{Ours}_{\text{BayElite}}$		✓	✓	81.04 ± 11.00
Ours*	✓	✓	✓	$\mathbf{88.99 \pm 9.60}$

(b) Different state augmentation and rewards.

State aug. / Reward	Test(%)
t+flags / STL	11.73 ± 2.67
t+flags / STL+Inv	46.85 ± 10.53
t+flags / STL+Inv+Prog	49.80 ± 7.72
t+flags / STL+Inv+Dist	84.59 ± 7.88
$\emptyset$ / STL+Inv+Dist	11.43 ± 3.48
t / STL+Inv+Dist	47.51 ± 7.86
Ours* (all / all)	$\mathbf{88.99 \pm 9.60}$

hurt the performance, likely due to the myopic exploration at the beginning of the training, which restricts the agent from seeking more promising assignments. Hence, we mix the two sources of the time variables together and witness certain improvement ($\text{Ours}_{\text{mixBay}}$, $\text{Ours}_{\text{mixElite}}$, and $\text{Ours}_{\text{BayElite}}$) compared to Ours. Finally, by combining all these together, Ours* achieves the best performance.

In Tbl. 1b we study how state augmentation and reward shaping foster an efficient multi-stage RL. For the reward design, we consider just using parts of the reward terms introduced before, and the results (the first 4 rows) show that, just using STL robustness score will only result in 11.73% success rate, whereas by gradually adding invariance penalty, progress reward and the distance reward, the performance will get improved (the most improvement comes from using the distance reward term) and finally becomes 88.99% for Ours*. Regarding the state augmentation, removing the flags in the augmented state will result in a 41.48% drop in success rate, and if further removing the time index counter, the performance will drop to 11.43%. The combined findings validate our design.

5.5 Visualization for interpretability and multi-modal behavior

TGPO can generate diverse behaviors to fulfill the STL specifications, which can also be reflected from the critic values. We consider the STL task $F_{[0,160]}(A_1) \wedge F_{[0,160]}(A_2) \wedge G_{[0,200]}\neg B$ in the Ant environment. The ant starts from the lower left, and there is an obstacle in the middle of the scene. The time variables here correspond to "Reach A_1" (the cyan region in the scene) and "Reach A_2" (green). After the training, we plot the critic value heatmap across different time variable assignments for the initial state. As shown in Fig. 7, the lower-left L-shape region is in low critic value as it is dynamically infeasible to reach the first subgoal in a short time ($0 \leq \tau \leq 40$). The diagonal line region also receives low critic value, because the two subgoal regions cannot be visited in such a short time. The diagonal line splits the promising time variable regions (yellow) into two parts, which result in two different ways to fulfill the task (as shown from the time-elapsed simulation plot on the right). This shows that time-conditioned TGPO can generate multi-modal solutions for the STL task.

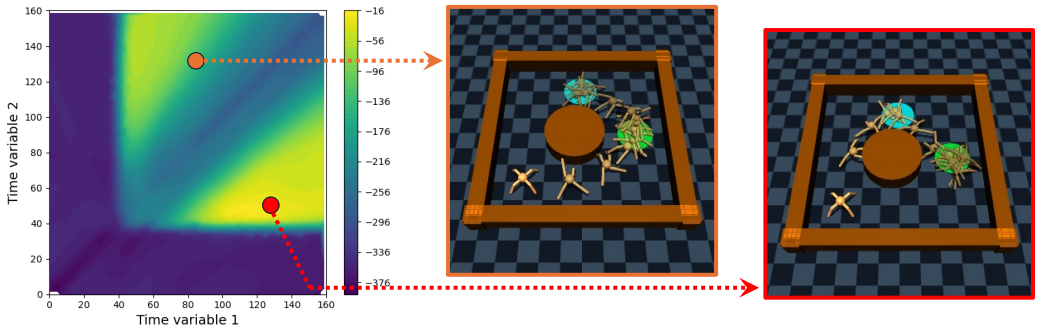


Figure 7: Critic value visualization and simulation for the ant environment under STL $F_{[0,160]}(A_1) \wedge F_{[0,160]}(A_2) \wedge G_{[0,200]}\neg B$. From the two feasible regions on the critic heatmap, we can see that the corresponding conditioned policy generates two behaviors to fulfill the task specification.

5.6 Limitations

While TGPO achieves strong empirical performance, it does not guarantee to converge to the global optimum, as is common for learning-based methods in complex nonconvex optimizations. TGPO is effective on STLs with conjunctions and temporal operators, but STLs with disjunctions or infinite-horizon tasks such as "Always-Eventually (G(F))" remain challenging in general and are not efficiently handled by existing approaches. To address these open challenges, we theoretically justify our reward design in App. A.2, provide preliminary results towards handling disjunctions (App. A.11) and scene generalization (App. A.12), and discuss about broader extensions and future directions in App. A.13.

6 Conclusion

In this paper, we introduce Temporal Grounded Policy Optimization (TGPO), a novel reinforcement learning framework for solving long-horizon Signal Temporal Logic tasks. By using STL decomposition, time variable sampling, state augmentation and reward design, TGPO can effectively handle complex STL and dynamics for long-horizon tasks. Experiments show that TGPO significantly outperforms existing baselines across various robotic environments and STL formulas. These results demonstrate TGPO as a promising paradigm for applying deep RL to STL tasks. Future work will focus on extending TGPO to handle a broader class of STL formulas and improving its scalability.

Acknowledgements

This work was supported by the National Science Foundation (NSF) under Grant #CCF-2238030. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and don't reflect the views of the sponsors.

References

[1] Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *55th Conference on Decision and Control (CDC)*, pages 6565–6570. IEEE, 2016.

[2] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial intelligence*, 116(1-2):123–191, 2000.

[3] Anand Balakrishnan and Jyotirmoy V Deshmukh. Structured reward shaping using signal temporal logic specifications. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3481–3486. IEEE, 2019.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: Composable transformations of python+ numpy programs, 2018.

[5] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. Decision-making with non-markovian rewards: From ltl to automata-based reward shaping. In *Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making*, pages 279–283, 2017.

[6] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[7] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[8] Alexandre Donzé. On signal temporal logic. In *Runtime Verification: 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings 4*, pages 382–383. Springer, 2013.

[9] Alexandre Donzé, Thomas Ferrere, and Oded Maler. Efficient robust monitoring for stl. In *Computer Aided Verification: 25th International Conference*, pages 264–279. Springer, 2013.

[10] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. *Formal Modeling and Analysis of Timed Systems*, page 92, 2010.

[11] Cameron Finucane, Gangyuan Jing, and Hadas Kress-Gazit. Ltlmop: Experimenting with language, temporal logic and robot control. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1988–1993. IEEE, 2010.

[12] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

[13] Yulong Gao, Alessandro Abate, Frank J Jiang, Mirco Giacobbe, Lihua Xie, and Karl Henrik Johansson. Temporal logic trees for model checking and control synthesis of uncertain discrete-time systems. *IEEE Transactions on Automatic Control*, 67(10):5071–5086, 2021.

[14] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.

[15] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*, pages 1–22. Springer, 2020.

[16] Yiting He, Peiran Liu, and Yiding Ji. Scalable signal temporal logic guided reinforcement learning via value function space optimization. *arXiv preprint arXiv:2408.01923*, 2024.

[17] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

[18] Junya Ikemoto and Toshimitsu Ushio. Deep reinforcement learning under signal temporal logic constraints using lagrangian relaxation. *IEEE Access*, 10:114814–114828, 2022.

[19] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Parv Kapoor, Anand Balakrishnan, and Jyotirmoy V Deshmukh. Model-based reinforcement learning from signal temporal logic specifications. *arXiv preprint arXiv:2011.04950*, 2020.

[21] Parv Kapoor, Eunsuk Kang, and Rômulo Meira-Góes. Safe planning through incremental decomposition of signal temporal logic specifications. In *NASA Formal Methods Symposium*, pages 377–396. Springer, 2024.

[22] Jesper Karlsson, Fernando S Barbosa, and Jana Tumova. Sampling-based motion planning with temporal logic missions and spatial preferences. *IFAC-PapersOnLine*, 53(2):15537–15543, 2020.

[23] Vincent Kurtz and Hai Lin. Mixed-integer programming for signal temporal logic with fewer binary variables. *IEEE Control Systems Letters*, 6:2635–2640, 2022.

[24] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, 42(6):356–370, 2023.

[25] Karen Leung and Marco Pavone. Semi-supervised trajectory-feedback controller synthesis for signal temporal logic specifications. In *2022 American Control Conference (ACC)*, pages 178–185. IEEE, 2022.

[26] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, 2017.

[27] Hsuan-Cheng Liao. A survey of reinforcement learning with temporal logic rewards. *preprint*, 2020.

[28] Alexis Linard, Ilaria Torre, Ermanno Bartoli, Alex Sleat, Iolanda Leite, and Jana Tumova. Real-time rrt* with signal temporal logic preferences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8621–8627, 2023.

[29] Ruijia Liu, Ancheng Hou, Xiao Yu, and Xiang Yin. Zero-shot trajectory planning for signal temporal logic tasks. *arXiv preprint arXiv:2501.13457*, 2025.

[30] Wenliang Liu, Noushin Mehdipour, and Calin Belta. Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints. *IEEE Control Systems Letters*, 6:91–96, 2021.

[31] Wenliang Liu, Wei Xiao, and Calin Belta. Learning robust and correct controllers from signal temporal logic specifications using barriernet. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 7049–7054. IEEE, 2023.

[32] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. *Formal Techniques, ModellingandAnalysis of Timed and Fault-Tolerant Systems*, page 152, 2004.

[33] Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. Arithmetic-geometric mean robustness for control from signal temporal logic specifications. In *2019 American Control Conference (ACC)*, pages 1690–1695. IEEE, 2019.

[34] Yue Meng and Chuchu Fan. Signal temporal logic neural predictive control. *IEEE Robotics and Automation Letters*, 8(11):7719–7726, 2023.

[35] Yue Meng and Chuchu Fan. Diverse controllable diffusion policy with signal temporal logic. *IEEE Robotics and Automation Letters*, 2024.

[36] Yue Meng and Chuchu Fan. Telograf: Temporal logic planning via graph-encoded flow matching. In *Forty-second International Conference on Machine Learning*, 2025.

[37] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1235–1240. IEEE, 2017.

[38] Aniruddh Puranic, Jyotirmoy Deshmukh, and Stefanos Nikolaidis. Learning from demonstrations using signal temporal logic. In *Conference on Robot Learning*, pages 2228–2242. PMLR, 2021.

[39] Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit A Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 1091–1096. IEEE, 2014.

[40] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[42] Mayank Sewlia, Christos K Verginis, and Dimos V Dimarogonas. Cooperative sampling-based motion planning under signal temporal logic specifications. In *2023 American Control Conference (ACC)*, pages 2697–2702. IEEE, 2023.

[43] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínskỳ. Limit-deterministic büchi automata for linear temporal logic. In *International Conference on Computer Aided Verification*, pages 312–332. Springer, 2016.

[44] Nikhil Kumar Singh and Indranil Saha. Stl-based synthesis of feedback controllers using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15118–15126, 2023.

[45] Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(2):3451–3458, 2022.

[46] Abdelhamid Tayebi and Stephen McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on control systems technology*, 14(3):562–571, 2006.

[47] Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A Mcilraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.

[48] Peter Varnai and Dimos V Dimarogonas. On robustness metrics for learning stl tasks. In *2020 American Control Conference (ACC)*, pages 5394–5399. IEEE, 2020.

[49] Cristian-Ioan Vasile, Vasumathi Raman, and Sertac Karaman. Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3840–3847. IEEE, 2017.

[50] Harish Venkataraman, Derya Aksaray, and Peter Seiler. Tractable reinforcement learning of signal temporal logic objectives. In *Learning for Dynamics and Control*, pages 308–317. PMLR, 2020.

[51] Siqi Wang, Xunyuan Yin, Shaoyuan Li, and Xiang Yin. Tractable reinforcement learning for signal temporal logic tasks with counterfactual experience replay. *IEEE Control Systems Letters*, 2024.

[52] Pian Yu, Xiao Tan, and Dimos V Dimarogonas. Continuous-time control synthesis under nested signal temporal logic specifications. *IEEE Transactions on Robotics*, 40:2272–2286, 2024.

A Appendix

A.1 Broader impacts

This paper presents work whose goal is to advance the field of machine learning and autonomous systems. From the autonomy standpoint, since the TGPO policy does not guarantee safety during deployment, an additional safety filter is required to prevent harm to humans or the system itself.

A.2 Theoretical analysis of reward consistency towards STL satisfaction

Here we first show that under the reward dominant assumption, the optimal policy that maximizes the expected return proposed in Eq. 6 is able to fulfill the task specification (Thm. A.1). In addition, with mild assumptions, we show that the optimal policy can also lead to satisfaction-probability maximization (Thm. A.2).

Theorem A.1 (Consistency of reward design and STL satisfaction)**.** *Assume the problem is feasible and consider a fixed initial state s_0. Let $Z(\tau) = \sum_{t=0}^{T} \gamma^t \mu(s_t)$ be the cumulative discounted "R_{dist}"-related return of the trajectory τ (we omit the subscripts for μ for brevity). Let $\Delta Z = \max_{\tau} Z(\tau) - \min_{\tau'} Z(\tau')$ be the maximum difference in Z between any two trajectories. If $\exists \lambda_1 > 0, \lambda_2 > 0, \lambda_3 > 0, \lambda_4 < 0$ such that:*

$$\gamma^T(N_g\lambda_2 + \lambda_3) - (N_g - 1)\lambda_2 > \lambda_1 \Delta Z,$$
$$\gamma^T(\lambda_3 - \lambda_4) - N_g\lambda_2(1 - \gamma^T) > \lambda_1 \Delta Z,$$

then the optimal policy of TGPO will satisfy the STL task.

Proof. To ensure the optimal policy satisfies the STL, the worst-case discounted return from the satisfiable trajectories should be higher than the best-case discounted return from non-satisfiable trajectories.

1. The return for the satisfiable trajectories $G_{sat} \geq \lambda_1 \min_{\tau} Z(\tau) + \lambda_2 N_g \gamma^T + \lambda_3 \gamma^T$, where the first part accounts for the reward-shaping, the second part is for progress reward (the worst-case will be all the goals are reached at very last steps), and the last is for the task success.

For the non-satisfiable trajectories, there are two possible cases: (1) failure due to not reaching all the goals, or failure due to violating the invariant properties.

2. The return for non-reaching-all-goal trajectories $G_{inc} \leq \lambda_1 \max_\tau Z(\tau) + \lambda_2(N_g - 1)$. (The highest return is achieved by visiting all but the last subgoals in the earliest possible time, and satisfying all the invariant constraints.)

3. The return for invariance-violation trajectories $G_{vio} \leq \lambda_1 \max_\tau Z(\tau) + \lambda_2 N_g + \lambda_4 \gamma^T$. (The highest return is achieved by visiting all the subgoals in the earliest possible time, and just violating one constraint in the last time step.)

To ensure the optimal policy solves the STL task, we should have $G_{sat} \geq \max(G_{inc}, G_{vio})$, thus:

$$\lambda_1 \min_\tau Z(\tau) + \lambda_2 N_g \gamma^T + \lambda_3 \gamma^T > \lambda_1 \max_\tau Z(\tau) + \lambda_2(N_g - 1),$$
$$\lambda_1 \min_\tau Z(\tau) + \lambda_2 N_g \gamma^T + \lambda_3 \gamma^T > \lambda_1 \max_\tau Z(\tau) + \lambda_2 N_g + \lambda_4 \gamma^T.$$

Therefore, we have

$$\gamma^T(N_g\lambda_2 + \lambda_3) - (N_g - 1)\lambda_2 > \Delta Z,$$
$$\gamma^T(\lambda_3 - \lambda_4) - N_g\lambda_2(1 - \gamma^T) > \Delta Z.$$

□

Next we show that with mild assumptions, the return maximization can lead to satisfaction-probability maximization.

Theorem A.2 (Consistency of reward design and STL satisfaction-probability maximization)**.** *For uniform initial state distribution. Given a fixed time assignment, denote $G_\pi(s)$ the return for state s under policy π. Assume $\forall \pi_1, \pi_2$, $\exists \pi_3$ s.t. $G_{\pi_3}(s) \geq \max\{G_{\pi_1}(s), G_{\pi_2}(s)\}$ for all initial states s. Denote π^* the policy that maximizes the expected return, and $\tilde{\pi}^*$ that maximizes the probability of STL satisfaction. Then $\pi^* = \tilde{\pi}^*$.*

Proof. For STL ϕ and policy π, denote $S_\pi = \{s | \sigma_\pi(s) \models \phi\}$ as all the initial states from where the policy can satisfy STL. The assumption implies $\forall \pi_1, \pi_2, \exists \pi_3$ s.t. $S_{\pi_3} \supseteq S_{\pi_1} \cup S_{\pi_2}$. If $\pi^* \neq \tilde{\pi}^*$, then $S_{\tilde{\pi}^*} \supseteq S_{\pi^*}$ (otherwise $\exists \bar{\pi} \neq \tilde{\pi}^*$ s.t. $S_{\bar{\pi}} \supseteq S_{\pi^*} \cup S_{\tilde{\pi}^*}$ and then $\tilde{\pi}^*$ does not maximize the satisfaction probability). Note that $\tilde{\pi}^*$ converts some violating trajectories under π^* into satisfiable trajectories while keeping the rest at least not lower in return. Thm. A.1 shows the proposed reward design guarantees that satisfiable trajectories always have higher return than violated trajectories, under uniform distribution it is easy to show $\tilde{\pi}^*$ has a higher or equal expected return than π^*, which only holds when $\tilde{\pi}^* = \pi^*$. □

Remark This policy-capacity assumption is reasonable, since one can in principle construct a stronger policy by combining multiple policy networks and selecting, for each state, the policy with the highest return.

A.3 STL decomposition algorithm

Algorithm 2 Symbolic STL Decomposition

Input: STL formula ϕ
Output: Subgoals Θ_g, invariant constraints Θ_c, and time variables $\mathbf{t}$
Initialize: $\Theta_g \leftarrow \emptyset, \Theta_c \leftarrow \emptyset, \mathbf{t} \leftarrow \emptyset$
call DECOMPOSE($\phi, 0$)
return $\Theta_g, \Theta_c, \mathbf{t}$

procedure DECOMPOSE(ψ, t_{ref})
if $\psi = \bigwedge_{i=1}^{K} \psi_i$ **then**
  for $i = 1$ **to** K **do**
    call DECOMPOSE(ψ_i, t_{ref})
  end for
else if $\psi = \mathbf{G}_{[a,b]}\varphi$ **then**
  if φ is of form $\neg\mu$ **then**
    $\Theta_c \leftarrow \Theta_c \cup \{\text{Inv}(\neg\mu, [t_{ref} + a, t_{ref} + b])\}$
  else
    $\Theta_g \leftarrow \Theta_g \cup \{\text{Inv}(\mu, [t_{ref} + a, t_{ref} + b])\}$
  end if
else if $\psi = \mathbf{F}_{[a,b]}\varphi$ **then**
  Create symbolic variable $\tau \in [a, b]$
  $\mathbf{t} \leftarrow \mathbf{t} \cup \{\tau\}$
  if φ is atomic proposition μ **then**
    $\Theta_g \leftarrow \Theta_g \cup \{\text{Reach}(\mu, [t_{ref} + \tau, t_{ref} + \tau])\}$
  else
    call DECOMPOSE($\varphi, t_{ref} + \tau$)
  end if
else if $\psi = \varphi_1 \mathbf{U}_{[a,b]}\varphi_2$ **then**
  Create symbolic variable $\tau \in [a, b]$
  $\mathbf{t} \leftarrow \mathbf{t} \cup \{\tau\}$
  if φ_2 is atomic proposition μ **then**
    $\Theta_g \leftarrow \Theta_g \cup \{\text{Reach}(\mu, [t_{ref} + \tau, t_{ref} + \tau])\}$
  else
    call DECOMPOSE($\varphi_2, t_{ref} + \tau$)
  end if
  if φ_1 is of form $\neg\mu$ **then**
    $\Theta_c \leftarrow \Theta_c \cup \{\text{Inv}(\neg\mu, [t_{ref} + a, t_{ref} + \tau])\}$
  else
    $\Theta_g \leftarrow \Theta_g \cup \{\text{Inv}(\mu, [t_{ref} + a, t_{ref} + \tau])\}$
  end if
end if
end procedure

A.4 Algorithm hyperparameters

All the main hyperparameters used during training are shown in Table 2.

A.5 Simulation environment details

In this paper, we conduct experiments on five simulation environments (Linear, Unicycle, Franka Panda, Quadrotor, and Ant). The first four environments were implemented in plain JAX code by writing out the system dynamics, whereas the last one was adopted from the Mujoco JAX implementation. Detailed implementations are listed as follows.

Table 2: Hyperparameters assignments used for training TGPO*.

Hyperparameter	**L**inear; **U**nicycle; **F**rankaPanda; **Q**uadrotor; **A**nt
Network hidden units	(512, 512, 512)
Optimizer	Adam
Learning rate	3×10^{-4}
Weight decay	0.1
Grad norm clip	0.5
Random seeds	1007,1008,1009,1010,1011,1012,1013
Batch size	512
Epochs	1000 (L, U); 2000 (F, A); 4000(Q)
Time steps T	100 (L, U, F, Q); 200 (A)
Time duration Δt	0.2 (L, U); 0.05 (F, A); 0.1 (Q)
Distance reward λ_1	0.5
Progress reward λ_2	20.0
Success reward λ_3	20.0
Invariance penalty λ_4	-3.0 (L, F, Q); -3.5 (U); -1.5 (A)
Number of MCMC steps N_{MCMC}	500
Number of warmup steps N_{warmup}	200
Number of MCMC chains M_{MCMC}	512
Ratio of Randomly-sampled time variables η_{mcmc}	0.5
Ratio of MCMC-sampled time variables $\eta_{uniform}$	0.4
Ratio of Elite time variables η_{elite}	0.1
Elite buffer size $\lvert\mathcal{B}\rvert$	512

A.5.1 Linear

We use a single-integrator dynamics model. The 2D state $(x, y)^T$ represents the 2D coordinates on a xy-plane, and the 2D control input $(v, w)^T$ reflects the velocities in these two directions. The system dynamics is described as:

$$\begin{cases} x_{t+1} = x_t + v_t \Delta t \\ y_{t+1} = y_t + w_t \Delta t \end{cases} \tag{7}$$

We set the time step duration $\Delta t = 0.2s$.

A.5.2 Unicycle

We use a car-like dynamics model. The 4D state $(x, y, \theta, v)^T$ represents the 2D coordinates on the xy-plane, the heading angle of the robot and the velocity of the robot, respectively. The 2D input $(\omega, a)^T$ represents the angular velocity and the acceleration. The system dynamics can be described as:

$$\begin{cases} x_{t+1} = x_t + v_t \cos(\theta_t) \Delta t \\ y_{t+1} = y_t + v_t \sin(\theta_t) \Delta t \\ \theta_{t+1} = \theta_t + \omega_t \Delta t \\ v_{t+1} = v_t + a_t \Delta t \end{cases} \tag{8}$$

We set the time step duration $\Delta t = 0.2s$. The control actuation is limited at $[-1rad/s, +1rad/s] \times [-4m/s^2, +4m/s^2]$. The scene layout is $[-5m, +5m] \times [-5m, +5m]$ on the xy-plane.

A.5.3 Franka Panda

We use a 7 DoF Franka Panda robot arm model to conduct the simulation. The 7D state $(\theta_1, \theta_2, ..., \theta_7)^T$ represents the angle for all the joints where θ_7 is for the end-effector joint. The 7D control input $(\omega_1, \omega_2, ..., \omega_7)^T$ represents the angular velocity for all the joints. The dynamics follows a simple single-integrator case: $\theta_{i,t+1} = \theta_{i,t} + \omega_{i,t} \Delta t,$ for $i = 1, 2, ..., 7$. We set the time step duration $\Delta t = 0.05s$.

A.5.4 Quadrotor

We use a full quadrotor dynamics model [46] to conduct the simulation. The 12D state $(x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z)^T$ represents the 3D coordinate $\mathbf{p} = (x, y, z)^T$, the velocity vector $\mathbf{v} = (v_x, v_y, v_z)^T$, the orientation vector $\mathbf{\Theta} = (\phi, \theta, \psi)^T$, and the angular velocity $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$, respectively. The 4D control input $(f_1, f_2, f_3, f_4)^T$ represents the lifting force from the four motors. The full dynamics are:

$$\begin{cases} \mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_t \Delta t \\ \mathbf{v}_{t+1} = \mathbf{v}_t + (g\mathbf{e}_3 - \frac{T}{m} R_z(\psi) R_y(\theta) R_x(\phi) \mathbf{e}_3) \Delta t \\ \mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t + \boldsymbol{\omega}_t \Delta t \\ \boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + I^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega}_t \times (I\boldsymbol{\omega}_t)) \Delta t \end{cases} \tag{9}$$

with the rotation matrices $R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$, $R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$, and $R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and T and $\boldsymbol{\tau}$ are the total thrust and the torques derived from the motor input u with the Coriolis effect considered to the angular velocity vector. We set the time step duration $\Delta t = 0.10s$, adapt the gravity coefficient $g = 9.81 m/s^2$ with the corresponding gravity vector $\mathbf{e}_3 = (0, 0, 1)^T$, set the total mass of the quadrotor $m = 0.2kg$ and set the diagonal line of the quadrotor inertia matrix I as $(0.01kg \cdot m^2, 0.01kg \cdot m^2, 0.02kg \cdot m^2)^T$.

A.5.5 Ant

In this case, the agent is a 8-DoF quadruped robot with the complex dynamics implemented in Brax [12]. The observation space is 29-dimension (3-dimension for xyz coordinates, 4-dimension for the torso orientation (in Quaternion representation), 3-dimension velocity vector and 3-dimension angular velocity for the torso, 8-dimension for the joints' angles and another 8-dimension for the joints' angular velocities). The original control input is 8-dimension for the torques applied to each of the 8 joints. To ease the RL training, we first train a goal-reaching policy, enabling the ant to learn and move to a specified target location. Then, for the baselines and our methods, the problem becomes planning the waypoints so that the ant can satisfy the STL tasks specified.

A.6 Baseline implementation details

A.6.1 CEM

We use the Cross Entropy Method baseline mentioned in [34], which belongs to the evolutionary search algorithm mentioned in [40]. We denote the initial neural network policy parameters as $\theta^{(0)}$. At j-th iteration, we draw N samples $\theta_1, ...\theta_N$ from $\mathcal{N}(\theta^{(j)}, {\sigma^{(j)}}^2)$ where $\sigma^{(j)}$ is the preset standard deviation, then we rollout the trajectories and compute their robustness score. We pick the top-K candidates parameters $\theta_{E_1}, ...\theta_{E_k}$. Then we update the estimate for the neural network parameters $\theta^{(j+1)} = \frac{1}{k} \sum_{i=1}^{k} \theta_{E_i}$ and $\sigma^{(j+1)} = \sqrt{\frac{1}{k-1} \sum_{i=1}^{k} (\theta_{E_i} - \theta^{(j+1)})^2}$. We repeat this process for L iterations to get the final parameters. We set the size for the elite pool to be $K = 32$ and set the population sample size to be $N = 512$. The number of iteration steps L is the same as our method ($L = 1000$ for "Linear" and "Unicycle", $L = 2000$ for "Franka Panda" and "Ant", and $L = 4000$ for "Quadrotor".)

A.6.2 τ-MDP

τ-MDP is an RL method introduced in [1] to solve STL tasks under the discrete state space. The original method appends history to the state space, and uses Q-learning to solve short-horizon tasks with 2-layer STL specifications. Here, we extend it to handle general STL formulas by augmenting the entire trajectory into the state space with STL robustness score as the terminal reward to guide the agent to satisfy STL tasks. We also changed the RL backbone from Q-learning to PPO for better

scalability to longer-horizon tasks (The original Q-learning tabular formulation will not work on continuous space for $T = 100$).

A.6.3 F-MDP

F-MDP is an improved RL method introduced in [50] to solve STL tasks under the discrete state space more efficiently. This approach considers the 2-layer STL specifications, and introduces a flag for each of the subformulas in the STL. They defined the state transition rules and reward mechanism for "F" and "G"-based subformulas based on these flags and show that the Q-learning under this augmentation can learn more efficiently than the Q-learning under τ-MDP [1]. We re-implemented F-MDP in PPO for our comparison.

A.6.4 RNN

In this case, similar to [30], we use an RNN to encode the history data and then use the robustness score as the final reward to guide the agent to satisfy the tasks. The issue of this implementation is that it is much more time-consuming compared to the other baselines.

A.6.5 Grad

In this case, similar to [25] and [34], we use a neural network policy to roll out the trajectory (in a deterministic manner, rather than sampling from the learned Gaussian distribution). At each time step, the network receives the state (and the time index) and generates the action, which is then sent to the environment to derive the next state. We repeat this process T times to roll out the full trajectory, which preserves the gradient through the differentiable system dynamics. We use the approximated robustness score mentioned in [37] to ensure the score is differentiable. We then conduct backpropagation-through-time (BPTT) to update the neural network parameters.

A.7 Temporal sampling algorithm details

The Metropolis-Hastings algorithm [6] is a Markov Chain Monte Carlo (MCMC) method for sampling from a probability distribution, commonly used when directly sampling from the distribution is hard. In our approach TGPO*, we use a discrete version of the M-H algorithm to sample time variables $\mathbf{t}$ that are likely to yield high critic values $V_\psi(s_0, \mathbf{t})$, where s_0 is the initial state. We use $\exp(V_\psi(s_0, \mathbf{t}))$ as a proxy for the unnormalized probability of the promising temporal variables. The algorithm proceeds by starting with an initial set of temporal variables $\mathbf{t}_0$ and iteratively proposing to move on grids to a new set $\mathbf{t}'$ based on a proposal distribution $g(\mathbf{t}'|\mathbf{t})$. The move is then accepted or rejected based on the acceptance ratio α, which compares the critic value exponentials of the new and the current variables. The process is detailed in Algorithm. 3.

Algorithm 3 Metropolis-Hastings for time variable sampling (with multiple chains and warm-up)

Input: Initial state s_0, Critic network $V_\psi(s, \mathbf{t})$, Proposal distribution $g(\mathbf{t}'|\mathbf{t})$
Input: Iterations N_{mcmc}, Number of chains M_{chain}, Number of warm-up steps N_{warmup}
for all chain $m \in \{1, \ldots, M_{chain}\}$ **do**
  Initialize temporal variables $\mathbf{t}_{m,0}$ randomly
  Initialize samples list $S_m \leftarrow []$
end for
for $i = 1$ **to** N_{mcmc} **do**
  for all chain $m \in \{1, \ldots, M_{chain}\}$ **do**
    // Propose new temporal variables for chain m
    $\mathbf{t}' \leftarrow$ Sample from $g(\mathbf{t}'|\mathbf{t}_{m,i-1})$
    // Calculate the acceptance ratio α
    $Q_{\text{current}} \leftarrow V_\psi(s_0, \mathbf{t}_{m,i-1})$
    $Q_{\text{new}} \leftarrow V_\psi(s_0, \mathbf{t}')$
    $\alpha \leftarrow \min(1, \exp(Q_{\text{new}} - Q_{\text{current}}))$
    // Accept or reject the new sample for chain m
    $u \leftarrow$ Sample from Uniform$(0, 1)$
    if $u < \alpha$ **then**
      $\mathbf{t}_{m,i} \leftarrow \mathbf{t}'$ // Accept the new sample
    else
      $\mathbf{t}_{m,i} \leftarrow \mathbf{t}_{m,i-1}$ // Reject and keep the old sample
    end if
  end for
end for
// Collect samples after the warm-up period
for $i = N_{warmup} + 1$ **to** N_{mcmc} **do**
  for all chain $m \in \{1, \ldots, M_{chain}\}$ **do**
    Add $\mathbf{t}_{m,i}$ to S_m
  end for
end for
Return: Sampled time variables $\{\mathbf{t} | \mathbf{t} \in S_m, m = 1, 2, \ldots, M_{chain}\}$

In our approach, we set $N_{mcmc} = 500$, $N_{warmup} = 200$, $M_{chain} = 512$ and pick the time variable from each S_i with the highest critic value to form the time variable set $\mathbf{T}_{mcmc}$ used in Alg. 1. For the proposal distribution $g(\mathbf{t}'|\mathbf{t})$, we use a uniform distribution over the local neighborhood of the current temporal variables $\mathbf{t}$: we first uniformly sample an index j from the dimensions of $\mathbf{t}$ and then uniformly sample a move direction $\Delta \in \{-1, +1\}$. The proposed new set of variables $\mathbf{t}'$ is generated by applying this change to the selected index but also ensure that the new value $\mathbf{t}'$ is within the valid range (otherwise we keep $\mathbf{t}$ unchanged).

A.8 Training time comparison

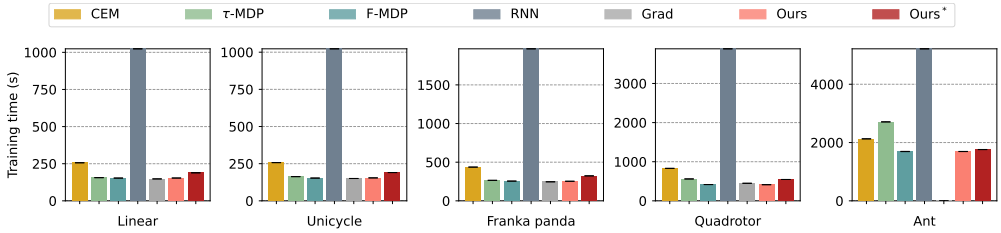


Figure 8: Comparison for training time. For each method under each environment, the result is averaged over 10 STLs and 7 random seeds. TGPO's training time is on par with leading baselines.

As shown in Fig. 8, TGPO and TGPO* have a similar runtime compared to τ-MDP, F-MDP and Grad baselines, whereas the CEM baseline is normally 20.8%$\sim$35.8% higher than TGPO*. The most time-consuming baseline is RNN, where TGPO* is 1.96X$\sim$6.11X faster in training speed. This

shows that our approach is as scalable as other top RL baselines in training time, but our method can achieve higher task success rate.

A.9 Comparison to classical solvers and reachability-based methods

Here we clarify the trade-off between exact control-synthesis and our proposed TGPO. Note that the classical approaches are inherently limited in scope: Mixed Integer Program (MIP) methods are only applicable to linear systems. Reachability-based methods rely on state-space discretization, thus they become increasingly expensive as the horizon length, STL nesting depth, or state dimension grows.

Mixed-integer Programming (MIP) baselines We compare in our "linear" benchmark under ten STL tasks with three MIP baselines: STL-INC [21], STLpy [23], and MA-STL [45]. STL-INC incrementally decomposes the STL task and solves the subtasks with MIP, STLpy solves the full STL task as a mixed-integer convex program with fewer binary variables, and MA-STL solves a sparse timed-waypoint MILP formulation. We set timeout to 600 seconds and monitor TGPO*'s first-success time, and report MIP runtime in two settings: first feasible solution time (closest comparison to TGPO*) and optimal solution time. For the first feasible solution time, as shown in Fig. 9, TGPO* is competitive with STL-INC and STLpy and successfully solves all ten tasks. At $n_t = 100$, TGPO* remains comparable to STLpy and is faster than STL-INC, while STLpy and MA-STL time out on the most difficult dense-planning task (Task ID $= 9$.) As the task horizon increases, TGPO* remains competitive to STLpy and faster than STL-INC. MA-STL is substantially faster than the rest approaches as it solves a sparse timed-waypoint formulation rather than dense per-step action sequences. Fig. 10 further shows that when the MIP baselines are required to compute optimal solutions, runtime increases substantially and several challenging settings time out, both for more difficult STL tasks and for longer horizons. This shows even in linear settings, exact STL optimization is substantially more expensive than finding a feasible trajectory, whereas TGPO* offers competitive feasible-solution runtime while being capable to extend to nonlinear and high-dimensional settings.

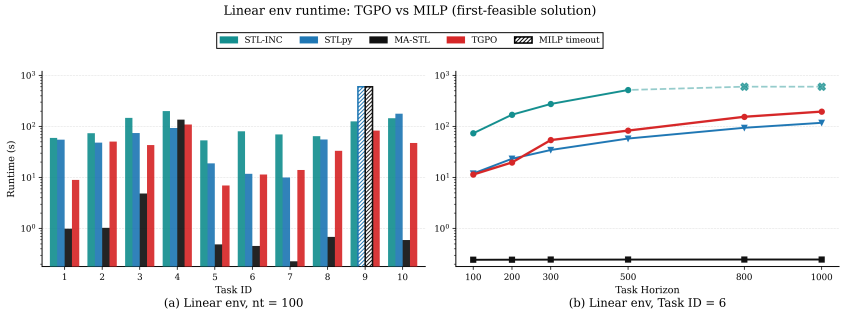


Figure 9: Runtime comparison between TGPO* and MILP-based baselines for first feasible solution time on linear STL tasks. TGPO* is competitive with STL-INC and STLpy across tasks and horizons, while some MILP baselines time out on harder instances.

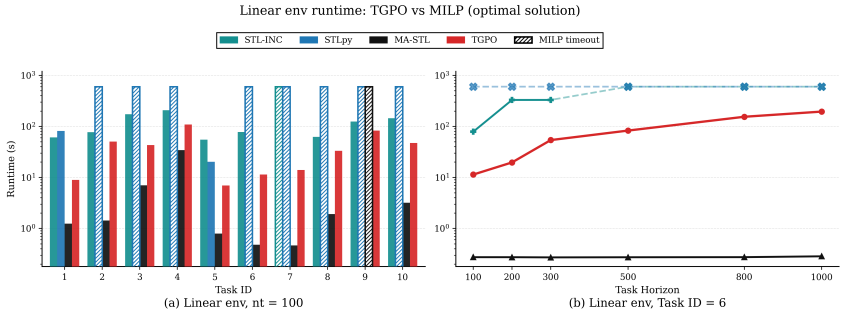


Figure 10: Runtime comparison when MILP baselines compute optimal solutions. Although this setting is not directly comparable to TGPO* first-success time, it shows that exact STL optimization is substantially more expensive and often times out on harder tasks or longer horizons.

Reachability-based baselines We consider the temporal logic tree (TLT) method in [13] under two low-dimensional systems ("Linear" and "Unicycle".) Concretely, we construct a controlled transition system by discretizing the state and action spaces, compute the controlled reachable set and robust controlled invariant set associated with the decomposed temporal tasks, and compile each STL task into an augmented state (x, p, t), where x is the system state, p is the progress state encoding the remaining temporal subgoals, and t is the time index. The controller is then synthesized by dynamic programming over this augmented transition system, selecting controls whose successors remain inside the corresponding feasible backward-reachable sets. For fair comparison, both the TLT baseline and TGPO are JAX-compiled and executed on a GPU. As shown in Fig. 11. On simple 2D linear tasks, the TLT baseline is often much faster than TGPO, showing that explicit reachability analysis is highly effective when the dynamics are low-dimensional and the discretization remains manageable. However, when tasks become more temporally nested, have longer outer temporal horizons, or involve higher-dimensional dynamics such as the 4D unicycle system, TLT becomes much harder to scale, leading to substantially increased runtime, timeout, or out-of-memory failures. By contrast, TGPO maintains a relatively stable training time and can solve all the tasks.

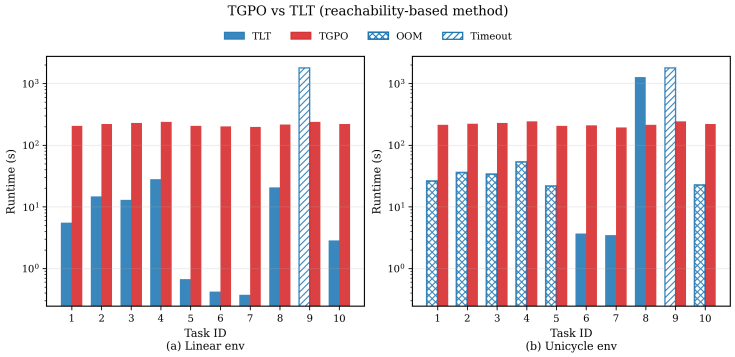


Figure 11: Runtime comparison between TGPO and a reachability-based method TLT under linear and unicycle dynamics. TLT is much faster on simple low-dimensional cases, but scales poorly on more temporally complex or higher-dimensional tasks where TGPO remains applicable.

Overall, these results support a complementary view. TGPO is not intended to replace guaranteed MIP or reachability-based synthesis methods on small low-dimensional linear problems, where formal methods can be highly effective and often faster. Rather, TGPO aims to provide a practical policy-learning framework for broader settings, especially nonlinear, high-dimensional, or black-box systems, where exact formal synthesis becomes increasingly difficult to apply directly.

A.10 Comparison to aggregation-based baselines

In recent years, aggregation semantics of STL have been widely used for reward generation in RL. Representative works include [33, 48, 44]. The comparisons against these methods are shown below. All experiments are conducted similarly to the main result (i.e., the task success rate is averaged over all ten STL specifications) in Fig. 4, with a single random seed 1007.

Here for [33] we replace the robustness score in our Grad baseline with the AGM score (so the Ant cell is blank, similar to Grad's case); for [48] we use their shadowing-robust metric with our CEM baseline, so that it is compared directly against the same optimiser; for [44] we use their high-level framework, in which the robustness of an online monitor is used as the RL reward, with PPO as the learner for fair comparison with our approach (we first tried SAC as in their paper, but it produced 0.000 everywhere, so we switched the RL component to PPO).

As shown in Tbl. 3, the aggregation-based metrics of [33] and [48] behave much like the standard robustness score: [33] gives 0.391/0.393/0.577/0.107 against Grad's 0.400/0.300/0.650/0.100, and [48] gives 0.235/0.116/0.000/0.000/0.033 against CEM's 0.304/0.132/0.008/0.000/0.025. This is within expectation. Both metrics are designed to agree with standard robustness on whether a specification is satisfied - they keep the sign of the score - and differ only in providing a smoother, better-shaped gradient in between, since standard min/max is determined entirely by the single worst timestep. That smoothing is a genuine contribution for the control-synthesis problems those papers

Table 3: Performance comparison with aggregation-based baselines.

Method	Linear	Unicycle	Panda	Quadrotor	Ant
TGPO* (ours)	**0.999**	**0.860**	**0.749**	**0.839**	**0.591**
RNN	0.497	0.212	0.297	0.125	0.204
Grad	0.400	0.300	0.650	0.100	–
CEM	0.304	0.132	0.008	0.000	0.025
Mehdipour et al. [33]	0.391	0.393	0.577	0.107	–
Varnai et al. [48]	0.235	0.116	0.000	0.000	0.033
Singh et al. [44]	0.584	0.588	0.361	0.000	0.296

target, but it is orthogonal to the difficulty in our benchmark: the bottleneck in nested long-horizon STL is deciding when each sub-goal must be met so that the remaining ones are still satisfiable. A smoother conjunction does not supply that information, which is why every aggregation we tested remains far below TGPO*.

For [44] we evaluated both robustness variants (classical, SSS) that their implementation supports. The smoothed aggregation (SSS) their paper proposes scores 0.000 on all fifty environment-specification cells, including the twenty-five in which the classical-robustness variant solves the task, and it stays at 0.000 across a 100x sweep of the safety weighting and a 16x increase in training budget. We therefore only report the classical variant for [44] in the table, which is stronger than the SSS variant on our benchmark. We verified our port line by line against their released code, matched it numerically to 1e-4 on constructed signals, reproduced their behaviour by running their own code and monitor on our environment, and repaired a numerical overflow in the reference implementation (its exponentials overflow at the margin magnitudes our scenes produce), without which that variant cannot be run on our benchmark at all.

The aggregation semantics of [33] and [48] perform on par with the standard-robustness baselines (Grad and CEM), and [44] performs comparably to our RNN baseline, while TGPO* still outperforms the best of them by 0.17–0.73 absolute success rate across the five environments. This clarifies our contribution: it isolates why nested STL tasks are hard to learn, namely temporal grounding rather than the smoothness or the aggregation of the robustness function, which is precisely the gap TGPO is designed to close.

A.11 Handling disjunction operators

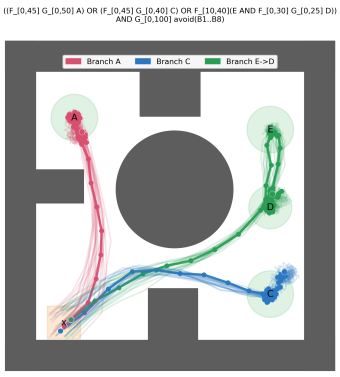


Figure 12: Illustration of handling STL formulae with disjunctions using TGPO* with additional variables. As discussed in Sec. 4.1, we can introduce binary (or integer-valued) variables in the augmented MDP to select the active branch of the disjunction and induce the corresponding reward path. In this example, TGPO* successfully learns trajectories satisfying the STL task, and different integer assignments select different branches, namely A, C, and E→D.

As mentioned in Sec. 4.1, the general framework of TGPO can handle STL with disjunctions by using extra variables. To justify this, we conduct an experiment under linear dynamic for the STL

spec $((F_{[0,45]}G_{[0,50]}A) \vee (F_{[0,45]}G_{[0,40]}C) \vee F_{[10,40]}(E \wedge F_{[0,30]}G_{[0,25]}D)) \wedge G_{[0,100]}\neg B_1 \wedge ... \wedge G_{[0,100]}\neg B_8$ where B_i represents the obstacles. In training we sample the branching variable and together with the time variables they induce the corresponding rewards. As shown in Fig. 12, the learned policy can satisfy the STL specification under all three branches. Handling infinite-horizon STL generally requires additional machinery for persistent satisfaction. For example for $G_{[0,\infty]}A$ we cannot simply check the trace, but rather we need to identify a set that keeps A satisfied (forward-invariance) and ensure the system starts within that set. This is outside the scope of our paper, and we view this as a direction for future extension.

A.12 Generalization and robustness under different environment layouts



Figure 13: A single TGPO policy trained with varied environment configurations on a fixed STL task. Left: STL and environment (the shaded region shows varied configurations: goal locations, sizes, shapes and obstacle locations.) Middle: the in-distribution evaluation shows an STL satisfaction rate of 0.996. Right: Out-of-Distribution (OOD) evaluation under shifts in goal regions, obstacles, and both, with satisfaction rates 0.906, 0.129, and 0.000, respectively (the green/red shading indicates the policy satisfies/violates the STL spec.) This shows one TGPO policy can generalize across varied in-distribution configurations, and certain degree of OOD scenarios.

In the main paper, the TGPO policy is trained for a specific STL under a fixed environment. Here we show that, with minor modifications, TGPO can learn to satisfy an STL specification under different environment configurations. For the STL task $F_{[10,30]}(C \wedge F_{[0,50]}G_{[0,9]}(A) \wedge G_{[0,99]}\text{Avoid(Gray)})$ under linear dynamics system, we consider different environment layouts (the location of the goal / obstacle regions and the shape and size of the regions) and send the parametrized scene descriptor to the TGPO policy (and critic) as an additional input. In training, we randomly sample different environment configurations, derive the corresponding reward function and transitions, and keep the rest training paradigm unchanged. As shown in Fig. 13, the learned policy on in-distribution evaluation shows an STL satisfaction rate of 0.996. For Out-of-Distribution (OOD) evaluation under shifts in goal regions, obstacles, and both, the learned policy achieves satisfaction rates 0.906, 0.129, and 0.000, respectively. This shows that under minor modifications, one TGPO policy can generalize across varied in-distribution configurations, and certain degree of OOD scenarios.

A.13 Blue-sky solutions regarding the current TGPO limitations

Here we discuss two directions to address the limitations mentioned in Sec. 5.6 for future work:

LLM-based logic weighting / pruning. Right now, all the time variable assignments are treated equally on the same level. To better handle (multiple) disjunction(s), we propose leveraging Large Language Models (LLMs) as a high-level logical prior. An LLM can reason over the semantic structure of the STL formula to weigh or prune unlikely branches of the logical graph and select the most promising disjunctive branch. This would reduce the problem from sampling over all logical possibilities to a single (or a subset of) high-likelihood branches.

Time variable-event binding for solution sparsity. A key limitation in scaling to the "Always Eventually" (GF) operators is the redundancy of time variables. For an STL like $G_{[0,200]}F_{[10,30]}\text{Reach(A)}$ a naive STL decomposition algorithm assigns a distinct time variable for every time step $t \in [0, 200]$,

but in fact these time variables are highly correlated. A single “Reach(A)" event at $t = 35$ can be used for all time variables $t \in [5, 25]$ for the outer “G" operator. Future work might consider working on a sparse set of K event anchors (or a partition for the time axis), where each anchor (partition) can bind with multiple time variables for the outer operator, which will effectively narrow the assignment complexity towards the number of sparse events. Constructing the event space and establishing the association between the event and time variables is left as an open question, with potential solutions including a learnable attention mechanism, hand-crafted and heuristic-based temporal clustering, or using an LLM to provide suggestions.

A.14 Solving STL with varied horizon lengths for the unicycle environment.

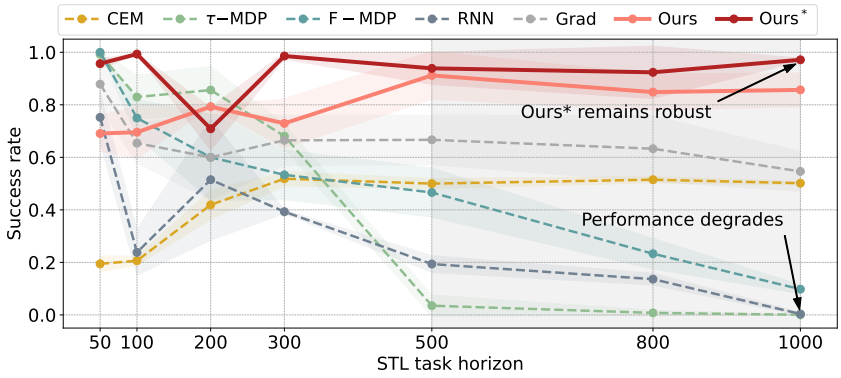
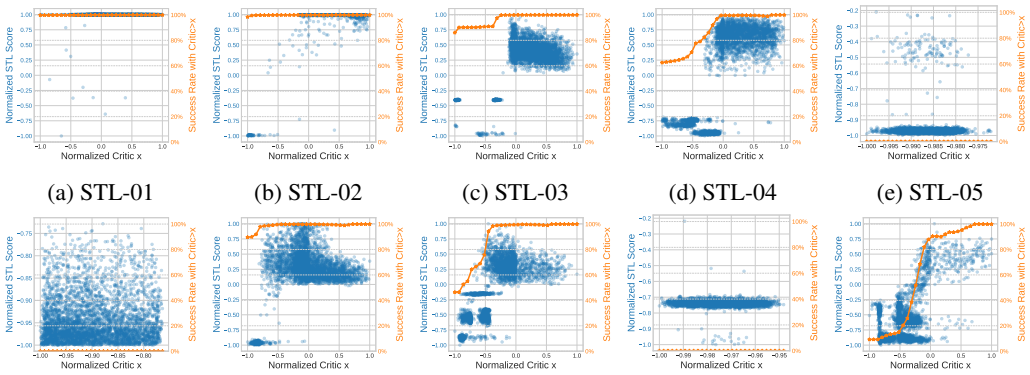


Figure 14: Solving STL in “Unicycle" environment over varied task horizons. Here we train each model with 3 random seeds. Similar to the trend in Fig. 6, our methods (TGPO and TGPO*) have the leading performance in T=100 case, and maintain high success rate in long horizons where RL baselines performance degrades.

We conduct experiments on the nonlinear “unicycle" environment to compare performance under varied task horizons (similar to Sec. 5.3, the temporal operator intervals are scaled accordingly). As shown in Fig. 14, our approaches (TGPO and TGPO*) have the competitive performance for short horizon settings (T=50,100), and maintain high success rate in longer horizons (T=300,500,800,1000) whereas other RL baselines encounter a degradation in the task success rate. The trend is similar to the one conducted under the “linear" environment (shown in Fig. 6), which showcases our approach’s effectiveness in handling long-horizon tasks is consistent across different system dynamics.

A.15 Correlation between the Critic and the STL robustness score

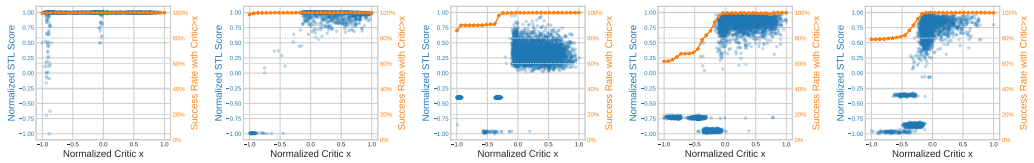


(a) STL-01 (b) STL-02 (c) STL-03 (d) STL-04 (e) STL-05

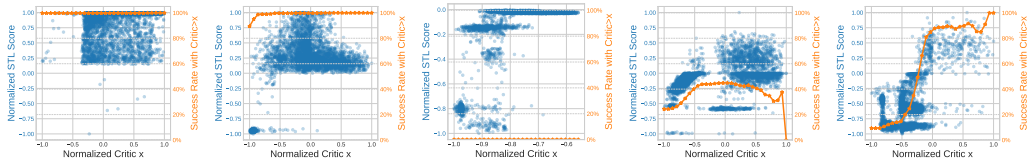
(f) STL-06 (g) STL-07 (h) STL-08 (i) STL-09 (j) STL-10

Figure 15: Correlation analysis between the critic and the STL score (seed=1007).

To validate that our learned critic in TGPO can really reflect the “promising" time variables that lead to STL satisfaction, in the “Linear" environment, for the TGPO algorithm, we randomly sample



(a) STL-01 (b) STL-02 (c) STL-03 (d) STL-04 (e) STL-05



(f) STL-06 (g) STL-07 (h) STL-08 (i) STL-09 (j) STL-10

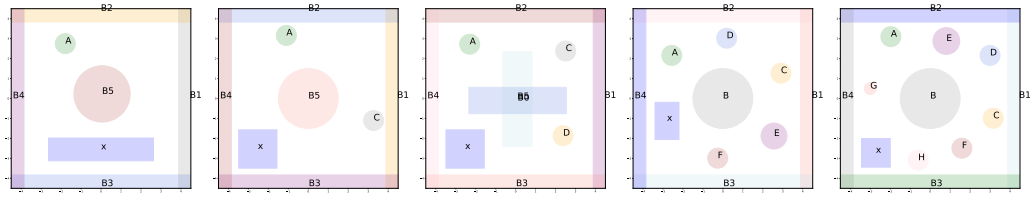
Figure 16: Correlation analysis between the critic and the STL score (seed=1008).

4096 points from the pretrained critic and rollout the corresponding trajectories to generate the STL robustness score. We plot the (critic value, STL score) scatter plot, together with the cumulative STL success rate curve for samples with a critic value greater than x. As shown in Fig. 15 (for seed=1007) and Fig. 16 (for seed=1008) from the blue scatter plots, whenever the critic value (left) is higher, the STL score is more likely to be higher, and hence more likely to satisfy STL. If we look at the orange curve, as the Critic value x increases, in most cases the probability for the corresponding traces satisfying the STL score is monotonously increasing or plateau at 100%, which indicates that our critic is learned correctly (note that if the critic is not learned well, it could learn for some time variables that bring in high critic value but result in low STL scores, like STL-09 in Fig. 16) and can be used to find "promising" time variable assignments.

A.16 STL Task details

Under each simulation environment, we make 10 STL formulas in two different categories ("two-level" and "multi-level"). Here we only consider predicates related with "Reach", "Stay", "Avoid" certain objects in the scene. They are listed as follows.

A.16.1 STLs in "Linear" environment



(a) STL-01 (b) STL-02 (c) STL-03 (d) STL-04 (e) STL-05

Figure 17: Scene for Linear: STL tasks 01 to 05

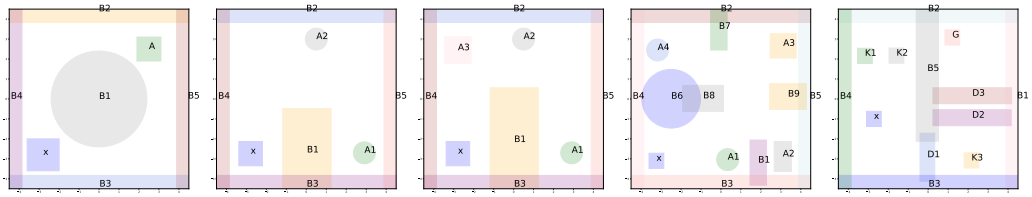
STL-01 (Two-layer): $F_{[5:7]}(F_{[50:85]}(A) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-02 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge G_{[60:80]}(C) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-03 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:60]}(C) \wedge G_{[70:80]}(D) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_0) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-04 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:50]}(C) \wedge F_{[70:80]}(F) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-05 (Two-layer): $F_{[5:10]}(F_{[0:30]}(A) \wedge F_{[30:50]}(C) \wedge F_{[70:80]}(F) \wedge F_{[75:88]}(H) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:90]}(\neg G) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$



(a) STL-06 (b) STL-07 (c) STL-08 (d) STL-09 (e) STL-10

Figure 18: Scene for Linear: STL tasks 06 to 10

STL-06 (Multi-layer): $F_{[10:90]}(A) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

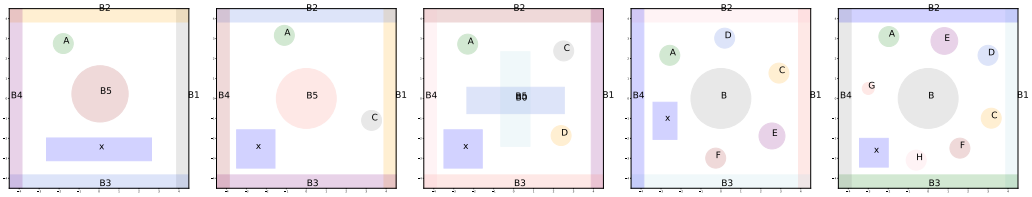
STL-07 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:80]}(A_2) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

STL-08 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:80]}(A_2 \wedge F_{[10:20]}(G_{[0:10]}(A_3))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

STL-09 (Multi-layer): $F_{[5:20]}(A_1 \wedge F_{[10:20]}(G_{[0:5]}(A_2) \wedge F_{[10:30]}(G_{[0:5]}(A_3)) \wedge F_{[10:30]}(G_{[0:10]}(A_4)))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5) \wedge G_{[0:100]}(\neg B_6) \wedge G_{[0:100]}(\neg B_7) \wedge G_{[0:100]}(\neg B_8) \wedge G_{[0:100]}(\neg B_9)$

STL-10 (Multi-layer): $(\neg D_1)U_{[0:100]}(K_1) \wedge (\neg D_2)U_{[0:100]}(K_2) \wedge (\neg D_3)U_{[0:100]}(K_3) \wedge F_{[80:90]}(G_{[0:5]}(G)) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

A.16.2 STLs in “Unicycle" environment



(a) STL-01 (b) STL-02 (c) STL-03 (d) STL-04 (e) STL-05

Figure 19: Scene for Unicycle: STL tasks 01 to 05

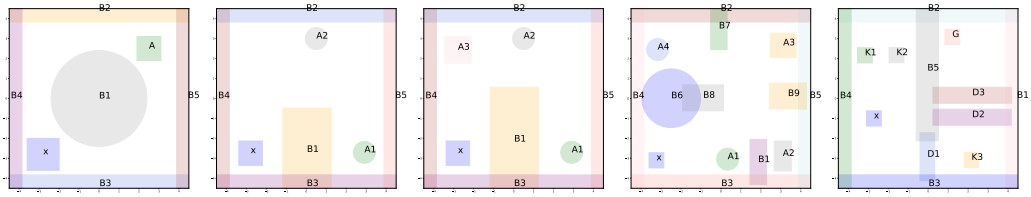
STL-01 (Two-layer): $F_{[5:7]}(F_{[50:85]}(A) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-02 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge G_{[60:80]}(C) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-03 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:60]}(C) \wedge G_{[70:80]}(D) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_0) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-04 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:50]}(C) \wedge F_{[70:80]}(F) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$

STL-05 (Two-layer): $F_{[5:10]}(F_{[0:30]}(A) \wedge F_{[30:50]}(C) \wedge F_{[70:80]}(F) \wedge F_{[75:88]}(H) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:90]}(\neg G) \wedge G_{[0:90]}(\neg B_1) \wedge G_{[0:90]}(\neg B_2) \wedge G_{[0:90]}(\neg B_3) \wedge G_{[0:90]}(\neg B_4))$



(a) STL-06 (b) STL-07 (c) STL-08 (d) STL-09 (e) STL-10

Figure 20: Scene for Unicycle: STL tasks 06 to 10

STL-06 (Multi-layer): $F_{[10:90]}(A) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

STL-07 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:80]}(A_2) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

STL-08 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:80]}(A_2 \wedge F_{[10:20]}(G_{[0:10]}(A_3))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

STL-09 (Multi-layer): $F_{[5:20]}(A_1 \wedge F_{[10:20]}(G_{[0:5]}(A_2) \wedge F_{[10:30]}(G_{[0:5]}(A_3)) \wedge F_{[10:30]}(G_{[0:10]}(A_4)))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5) \wedge G_{[0:100]}(\neg B_6) \wedge G_{[0:100]}(\neg B_7) \wedge G_{[0:100]}(\neg B_8) \wedge G_{[0:100]}(\neg B_9)$

STL-10 (Multi-layer): $(\neg D_1) U_{[0:100]}(K_1) \wedge (\neg D_2) U_{[0:100]}(K_2) \wedge (\neg D_3) U_{[0:100]}(K_3) \wedge F_{[80:90]}(G_{[0:5]}(G)) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg B_5)$

A.16.3 STLs in “Franka Panda" environment

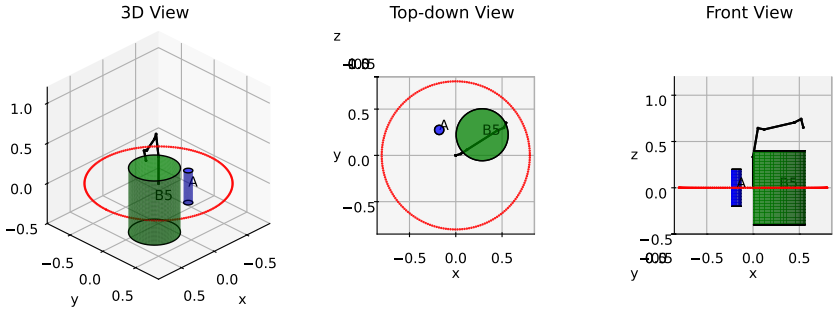


Figure 21: Scene for Franka Panda: STL task 01

STL-01 (Two-layer): $F_{[5:7]}(F_{[50:85]}(A) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

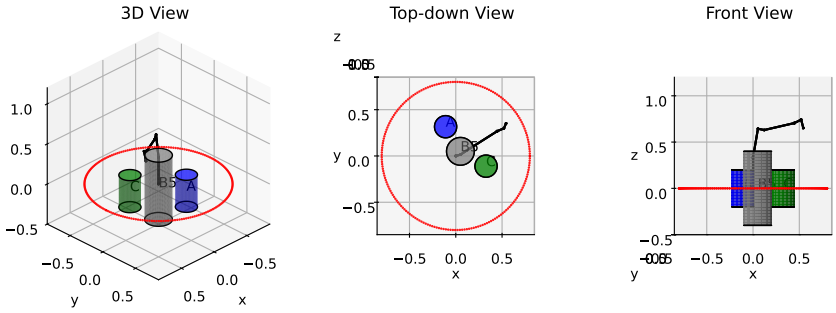


Figure 22: Scene for Franka Panda: STL task 02

STL-02 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge G_{[60:80]}(C) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

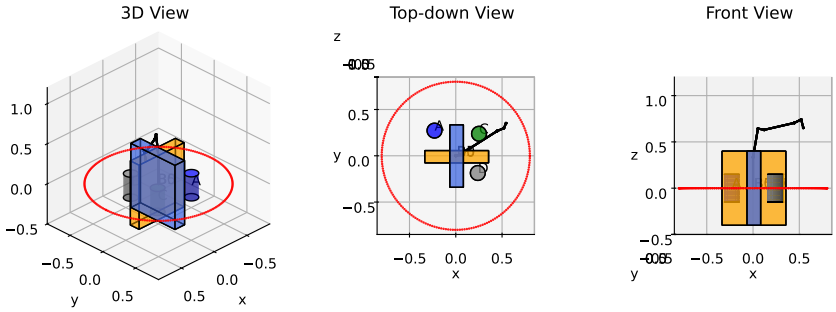


Figure 23: Scene for Franka Panda: STL task 03

STL-03 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:60]}(C) \wedge G_{[70:80]}(D) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_0) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

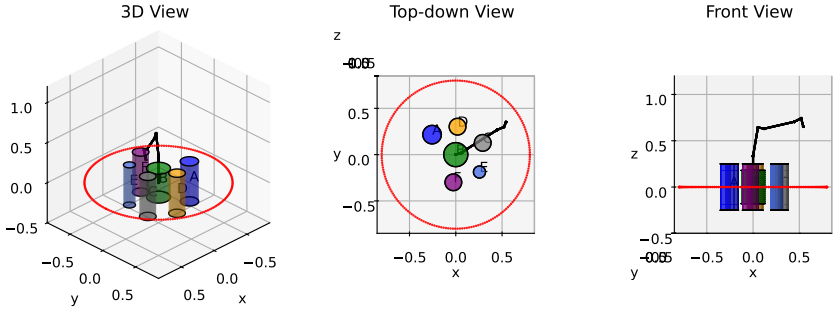


Figure 24: Scene for Franka Panda: STL task 04

STL-04 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:50]}(C) \wedge F_{[70:80]}(F) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

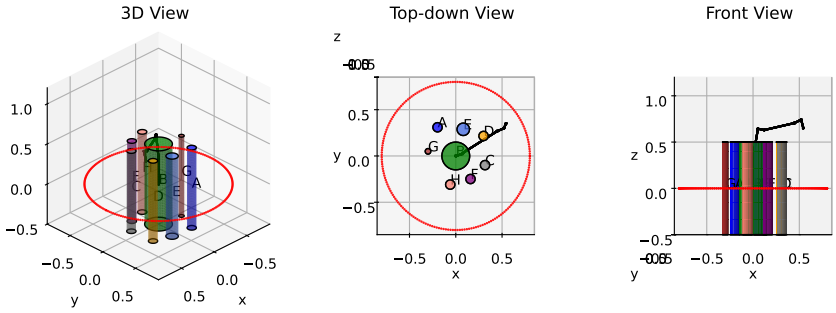


Figure 25: Scene for Franka Panda: STL task 05

STL-05 (Two-layer): $F_{[5:10]}(F_{[0:30]}(A) \wedge F_{[30:50]}(C) \wedge F_{[70:80]}(F) \wedge F_{[75:88]}(H) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:90]}(\neg G) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

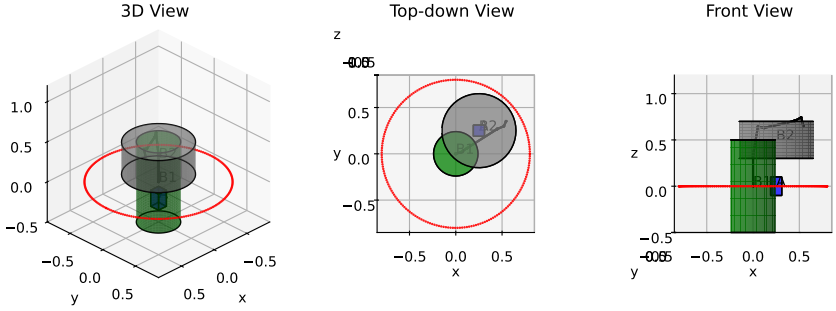


Figure 26: Scene for Franka Panda: STL task 06

STL-06 (Multi-layer): $F_{[10:90]}(A) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

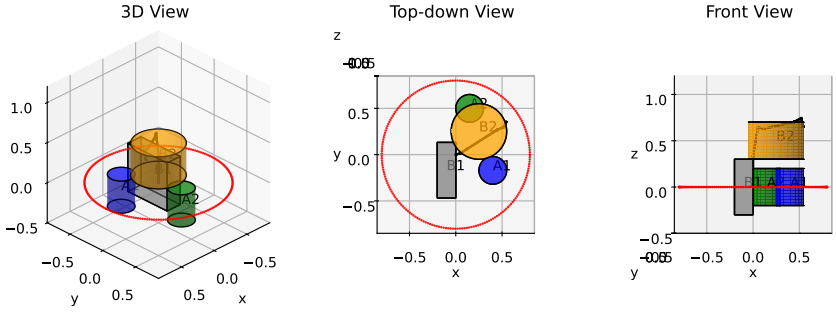


Figure 27: Scene for Franka Panda: STL task 07

STL-07 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:80]}(A_2) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

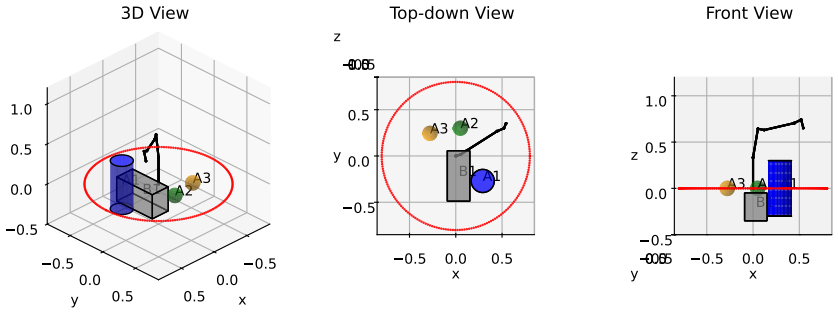


Figure 28: Scene for Franka Panda: STL task 08

STL-08 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:60]}(A_2 \wedge F_{[15:30]}(G_{[0:5]}(A_3))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

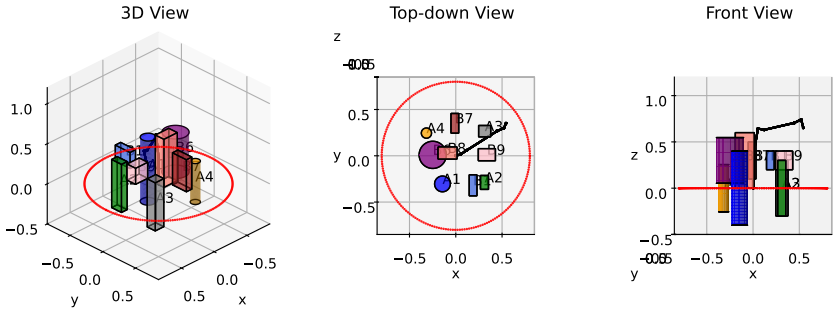


Figure 29: Scene for Franka Panda: STL task 09

STL-09 (Multi-layer): $F_{[25:30]}(A_1 \wedge F_{[20:28]}(G_{[0:5]}(A_2) \wedge F_{[10:30]}(G_{[0:5]}(A_3)) \wedge F_{[10:30]}(G_{[0:10]}(A_4)))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_6) \wedge G_{[0:100]}(\neg B_7) \wedge G_{[0:100]}(\neg B_8) \wedge G_{[0:100]}(\neg B_9) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

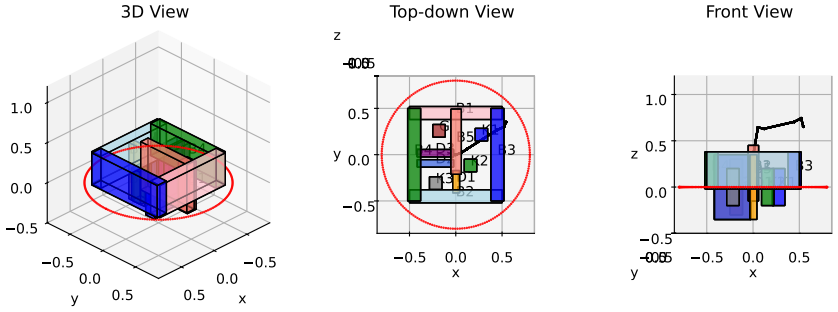


Figure 30: Scene for Franka Panda: STL task 10

STL-10 (Multi-layer): $(\neg D_1) U_{[0:100]}(K_1) \wedge (\neg D_2) U_{[0:100]}(K_2) \wedge (\neg D_3) U_{[0:100]}(K_3) \wedge F_{[80:90]}(G_{[0:5]}(G)) \wedge G_{[0:100]}(\neg B_5) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_2) \wedge G_{[0:100]}(\neg B_3) \wedge G_{[0:100]}(\neg B_4) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

A.16.4 STLs in “Quadrotor" environment

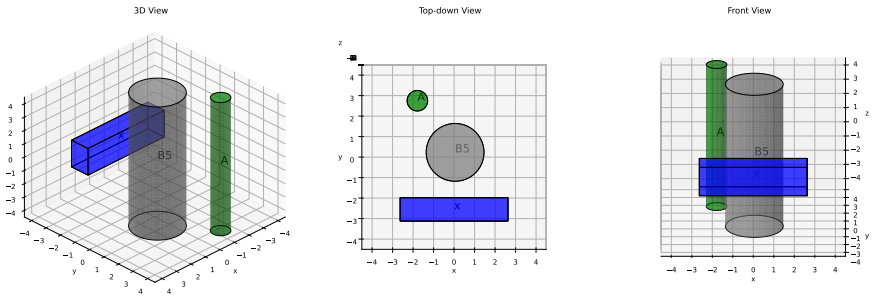


Figure 31: Scene for Quadrotor: STL task 01

STL-01 (Two-layer): $F_{[5:7]}(F_{[50:85]}(A) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

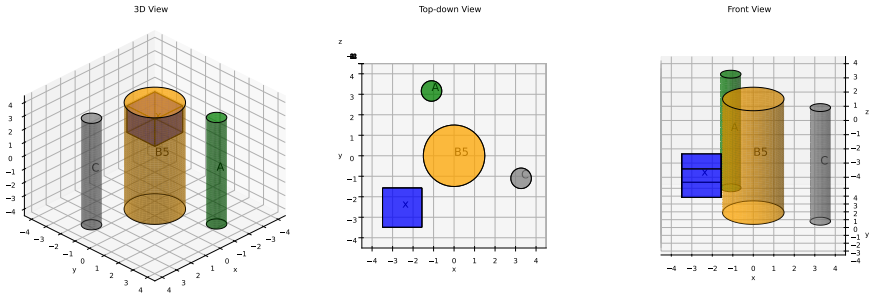


Figure 32: Scene for Quadrotor: STL task 02

STL-02 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge G_{[60:80]}(C) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

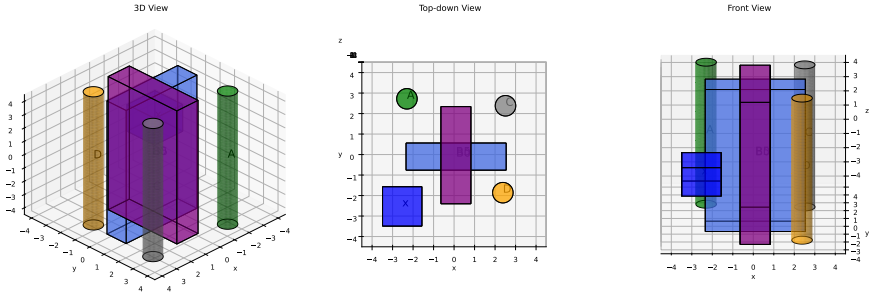


Figure 33: Scene for Quadrotor: STL task 03

STL-03 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:60]}(C) \wedge G_{[70:80]}(D) \wedge G_{[0:90]}(\neg B_5) \wedge G_{[0:90]}(\neg B_0) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

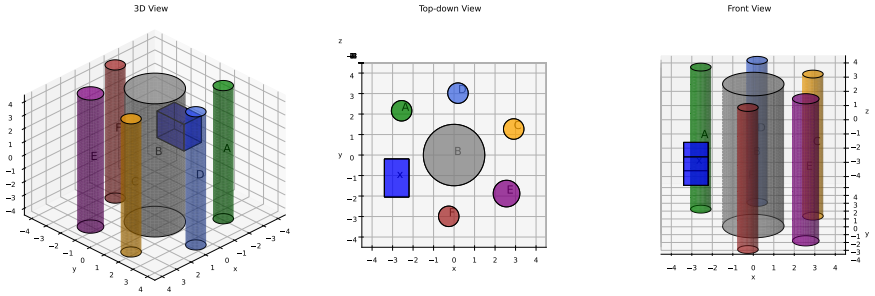


Figure 34: Scene for Quadrotor: STL task 04

STL-04 (Two-layer): $F_{[5:10]}(F_{[0:50]}(A) \wedge F_{[40:50]}(C) \wedge F_{[70:80]}(F) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

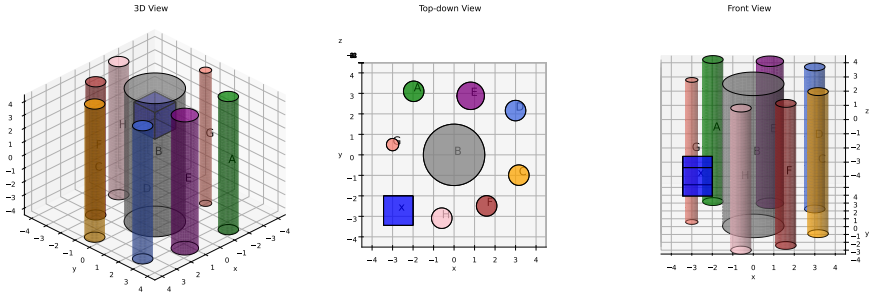


Figure 35: Scene for Quadrotor: STL task 05

STL-05 (Two-layer): $F_{[5:10]}(F_{[0:30]}(A) \wedge F_{[30:50]}(C) \wedge F_{[70:80]}(F) \wedge F_{[75:88]}(H) \wedge G_{[50:60]}(D) \wedge G_{[0:90]}(\neg B) \wedge G_{[0:90]}(\neg E) \wedge G_{[0:90]}(\neg G) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6))$

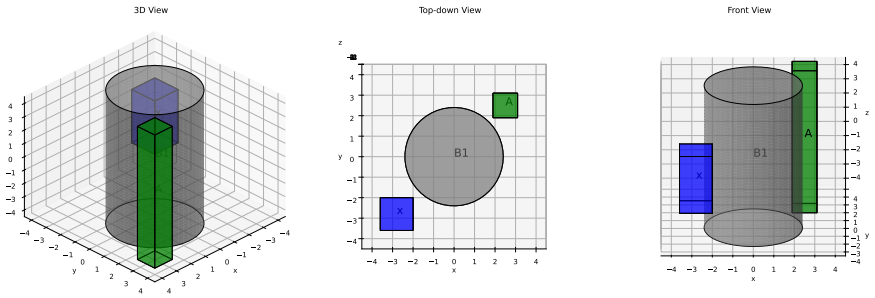


Figure 36: Scene for Quadrotor: STL task 06

STL-06 (Multi-layer): $F_{[10:90]}(A) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

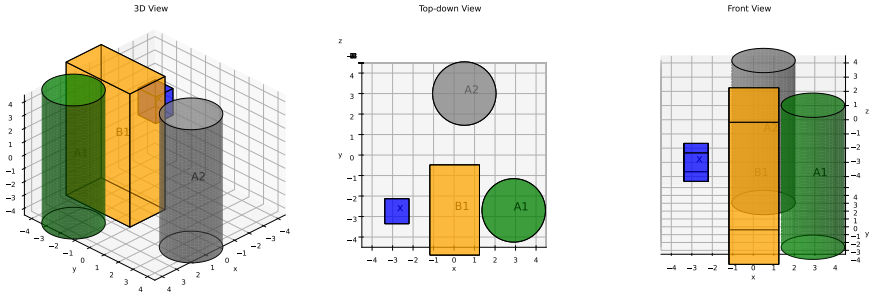


Figure 37: Scene for Quadrotor: STL task 07

STL-07 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:80]}(A_2) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

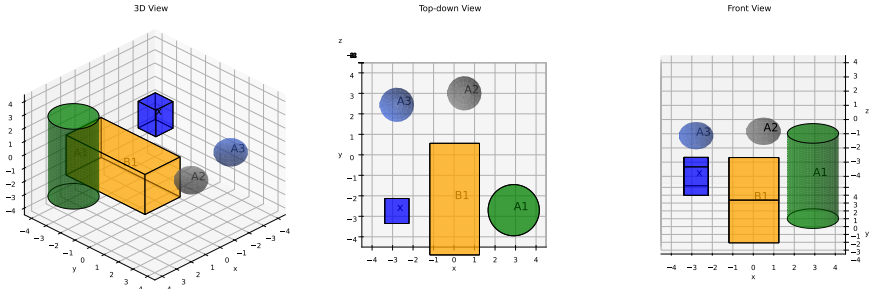


Figure 38: Scene for Quadrotor: STL task 08

STL-08 (Multi-layer): $F_{[0:90]}(A_1) \wedge F_{[40:60]}(A_2 \wedge F_{[15:30]}(G_{[0:5]}(A_3))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

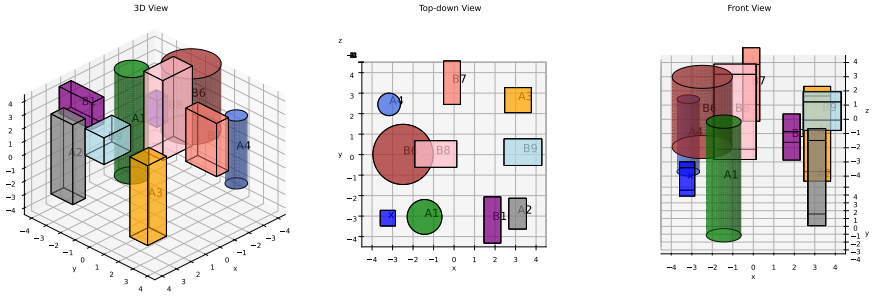


Figure 39: Scene for Quadrotor: STL task 09

STL-09 (Multi-layer): $F_{[25:30]}(A_1 \wedge F_{[20:28]}(G_{[0:5]}(A_2) \wedge F_{[10:30]}(G_{[0:5]}(A_3)) \wedge F_{[10:30]}(G_{[0:10]}(A_4)))) \wedge G_{[0:100]}(\neg B_1) \wedge G_{[0:100]}(\neg B_6) \wedge G_{[0:100]}(\neg B_7) \wedge G_{[0:100]}(\neg B_8) \wedge G_{[0:100]}(\neg B_9) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

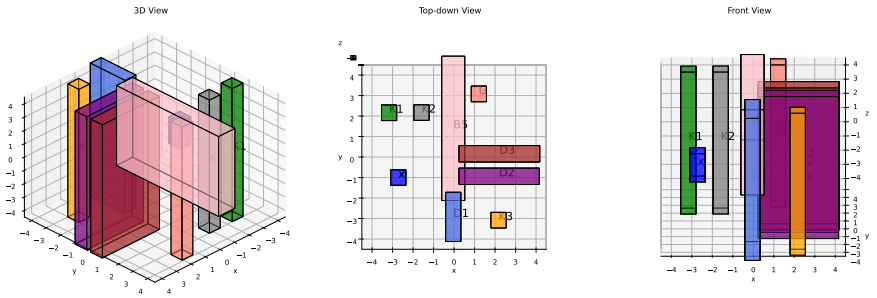
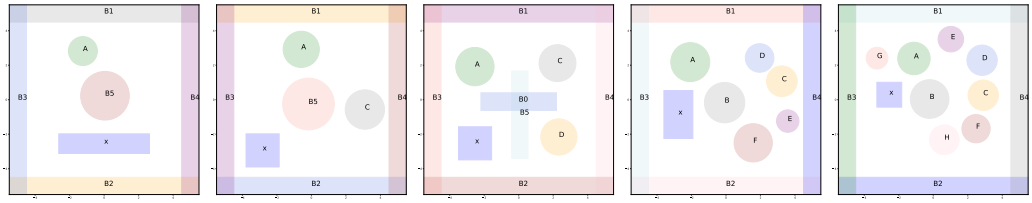


Figure 40: Scene for Quadrotor: STL task 10

STL-10 (Multi-layer): $(\neg D_1)U_{[0:100]}(K_1) \wedge (\neg D_2)U_{[0:100]}(K_2) \wedge (\neg D_3)U_{[0:100]}(K_3) \wedge F_{[80:90]}(G_{[0:5]}(G)) \wedge G_{[0:100]}(\neg B_5) \wedge G_{[0:100]}(\neg W_1) \wedge G_{[0:100]}(\neg W_2) \wedge G_{[0:100]}(\neg W_3) \wedge G_{[0:100]}(\neg W_4) \wedge G_{[0:100]}(\neg W_5) \wedge G_{[0:100]}(\neg W_6)$

A.16.5 STLs in "Ant" environment



(a) STL-01 (b) STL-02 (c) STL-03 (d) STL-04 (e) STL-05

Figure 41: Scene for Ant: STL tasks 01 to 05

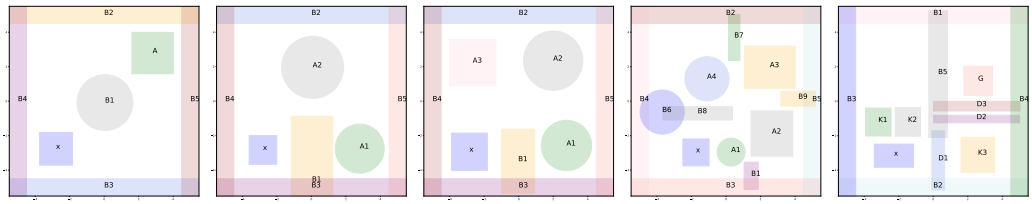
STL-01 (Two-layer): $F_{[10:14]}(F_{[100:170]}(A) \wedge G_{[0:180]}(\neg B_5) \wedge G_{[0:180]}(\neg B_1) \wedge G_{[0:180]}(\neg B_2) \wedge G_{[0:180]}(\neg B_3) \wedge G_{[0:180]}(\neg B_4))$

STL-02 (Two-layer): $F_{[10:20]}(F_{[0:100]}(A) \wedge G_{[120:160]}(C) \wedge G_{[0:180]}(\neg B_5) \wedge G_{[0:180]}(\neg B_1) \wedge G_{[0:180]}(\neg B_2) \wedge G_{[0:180]}(\neg B_3) \wedge G_{[0:180]}(\neg B_4))$

STL-03 (Two-layer): $F_{[10:20]}(F_{[0:100]}(A) \wedge F_{[80:120]}(C) \wedge G_{[140:160]}(D) \wedge G_{[0:180]}(\neg B_5) \wedge G_{[0:180]}(\neg B_0) \wedge G_{[0:180]}(\neg B_1) \wedge G_{[0:180]}(\neg B_2) \wedge G_{[0:180]}(\neg B_3) \wedge G_{[0:180]}(\neg B_4))$

STL-04 (Two-layer): $F_{[10:20]}(F_{[0:100]}(A) \wedge F_{[80:100]}(C) \wedge F_{[140:160]}(F) \wedge G_{[100:120]}(D) \wedge G_{[0:180]}(\neg B) \wedge G_{[0:180]}(\neg E) \wedge G_{[0:180]}(\neg B_1) \wedge G_{[0:180]}(\neg B_2) \wedge G_{[0:180]}(\neg B_3) \wedge G_{[0:180]}(\neg B_4))$

STL-05 (Two-layer): $F_{[10:20]}(F_{[0:60]}(A) \wedge F_{[60:100]}(C) \wedge F_{[140:160]}(F) \wedge F_{[150:176]}(H) \wedge G_{[100:120]}(D) \wedge G_{[0:180]}(\neg B) \wedge G_{[0:180]}(\neg E) \wedge G_{[0:180]}(\neg G) \wedge G_{[0:180]}(\neg B_1) \wedge G_{[0:180]}(\neg B_2) \wedge G_{[0:180]}(\neg B_3) \wedge G_{[0:180]}(\neg B_4))$



(a) STL-06 (b) STL-07 (c) STL-08 (d) STL-09 (e) STL-10

Figure 42: Scene for Ant: STL tasks 06 to 10

STL-06 (Multi-layer): $F_{[20:180]}(A) \wedge G_{[0:200]}(\neg B_1) \wedge G_{[0:200]}(\neg B_2) \wedge G_{[0:200]}(\neg B_3) \wedge G_{[0:200]}(\neg B_4) \wedge G_{[0:200]}(\neg B_5)$

STL-07 (Multi-layer): $F_{[0:180]}(A_1) \wedge F_{[80:160]}(A_2) \wedge G_{[0:200]}(\neg B_1) \wedge G_{[0:200]}(\neg B_2) \wedge G_{[0:200]}(\neg B_3) \wedge G_{[0:200]}(\neg B_4) \wedge G_{[0:200]}(\neg B_5)$

STL-08 (Multi-layer): $F_{[0:180]}(A_1) \wedge F_{[80:160]}(A_2 \wedge F_{[20:40]}(G_{[0:20]}(A_3))) \wedge G_{[0:200]}(\neg B_1) \wedge G_{[0:200]}(\neg B_2) \wedge G_{[0:200]}(\neg B_3) \wedge G_{[0:200]}(\neg B_4) \wedge G_{[0:200]}(\neg B_5)$

STL-09 (Multi-layer): $F_{[10:40]}(A_1 \wedge F_{[20:40]}(G_{[0:10]}(A_2) \wedge F_{[20:60]}(G_{[0:10]}(A_3)) \wedge F_{[20:60]}(G_{[0:20]}(A_4)))) \wedge G_{[0:200]}(\neg B_1) \wedge G_{[0:200]}(\neg B_2) \wedge G_{[0:200]}(\neg B_3) \wedge G_{[0:200]}(\neg B_4) \wedge G_{[0:200]}(\neg B_5) \wedge G_{[0:200]}(\neg B_6) \wedge G_{[0:200]}(\neg B_7) \wedge G_{[0:200]}(\neg B_8) \wedge G_{[0:200]}(\neg B_9)$

STL-10 (Multi-layer): $(\neg D_1)U_{[0:200]}(K_1) \wedge (\neg D_2)U_{[0:200]}(K_2) \wedge (\neg D_3)U_{[0:200]}(K_3) \wedge F_{[160:180]}(G_{[0:10]}(G)) \wedge G_{[0:200]}(\neg B_1) \wedge G_{[0:200]}(\neg B_2) \wedge G_{[0:200]}(\neg B_3) \wedge G_{[0:200]}(\neg B_4) \wedge G_{[0:200]}(\neg B_5)$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and / or introduction match the theoretical in App. A.2 and experimental results in Sec. 5. The results can be expected to generalize to other settings (disjunction cases in App. A.11 and scene-conditioned policy in App. A.12).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are presented in Sec. 5.6 and some potential solutions are discussed in App. A.13.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The theoretical results and the corresponding assumptions and a complete proof are shown in App. A.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All the hyperparameters used in the experiments are listed in the App. A.4 and the stochasticity in the experiments is controlled by explicit random seeds. The experimental results are reproducible.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code, including the proposed algorithm, simulations, and STL tasks, is publicly available at: `https://github.com/mengyuest/TGPO`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The highlevel description of the algorithm is shown in Sec. 4. The details of the approach is shown in the App. A.3 and App. A.7. All the hyperparameters used in the experiments are listed in the App. A.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports the error bars for the task success rate of the proposed method and baselines in Fig. 4, Fig. 5 and Fig. 6. The error bars are computed as the standard deviation of the metrics of interests under 7 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper presents the computer resource information in Sec. 5 and time of execution in App. A.8.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics after checking.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts have been discussed in App. A.1.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the assets (code, baseline methods) are properly credited in this manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset in this paper is well documented in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.