On the Hardness of Reinforcement Learning with Transition Look-Ahead

Corentin Pla
CREST, ENSAE
Criteo AI Lab
FairPlay Joint Team

Hugo Richard
Criteo AI Lab
FairPlay Joint Team

Marc Abeille
Criteo AI Lab
FairPlay Joint Team

Nadav Merlis
Technion

Vianney Perchet
CREST, ENSAE
Criteo AI Lab
FairPlay Joint Team

Abstract

We study reinforcement learning (RL) with transition look-ahead, where the agent may observe which states would be visited upon playing any sequence of ℓ actions before deciding its course of action. While such predictive information can drastically improve the achievable performance, we show that using this information optimally comes at a potentially prohibitive computational cost. Specifically, we prove that optimal planning with one-step look-ahead ($\ell = 1$) can be solved in polynomial time through a novel linear programming formulation. In contrast, for $\ell \geq 2$, the problem becomes NP-hard. Our results delineate a precise boundary between tractable and intractable cases for the problem of planning with transition look-ahead in reinforcement learning.

1 INTRODUCTION

Reinforcement Learning (RL) (Sutton and Barto, 2018) addresses the problem of learning how to act in a dynamic environment. This problem is modeled via a *Markov Decision Process* (MDP) which involves a *transition kernel*, describing how *states* of the environment evolve in response to the agent's *actions*, and a *reward function*, providing feedback to the agent for taking a particular action in a given state. The agent's goal is to select actions that maximize the cumulative collected reward called *return*, accounting not only for immediate gains but also for the long-term impact of its decisions on the state dynamics (Jaksch et al., 2010; Azar et al., 2017; Jin et al., 2018). In this work, we focus on *stationary* MDP, in which the reward function and transition kernel are independent of time.

In the standard RL framework, the reward and the next state are revealed only after an action has been taken. However, *RL with look-ahead* assumes that, in addition to this underlying dynamical model, extra predictive information is available at decision time. In *transition look-ahead*, the agent may observe before taking its action, which states would be visited upon playing any sequence of actions of length ℓ. This captures situations where one benefits from privileged information channels beyond standard interaction. A typical example is collaborative navigation systems that allow real-time traffic information (e.g., Waze, Coyote...) where information from nearby drivers can be used to estimate future position, speed, and traffic conditions given a sequence of routing decisions (Vasserman et al., 2015). Other examples include access to high-fidelity but expensive simulators that can provide look-ahead on demand, or supply-chain systems where estimated delivery or arrival times are provided in advance. Standard RL algorithms do not come with off-the-shelf tools to incorporate look-ahead, and a naive policy would be to just discard this additional information. By leveraging transition look-ahead, however, the agent can anticipate the consequences of its actions before execution, enabling more effective planning while reducing uncertainty about near-term dynamics.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

Related Work and Contribution. Our main result identifies a surprising complexity threshold: planning with one-step transition look-ahead ($\ell = 1$) is solvable in polynomial time, and we provide an explicit linear programming formulation. In stark contrast, planning with multi-step transition look-ahead ($\ell \geq 2$) is NP-hard. This establishes a sharp separation between tractable and intractable regimes, uncovering a fundamental frontier in the computational complexity of reinforcement learning with predictions.

The idea of augmenting reinforcement learning with look-ahead has recently begun to attract attention. Merlis (2024) introduced a pseudo-polynomial algorithm for planning with ℓ-transition look-ahead with $\ell = 1$ in the finite horizon setting. However, in stationary MDPs, there is no polynomial-time algorithm to solve planning in the finite horizon objective (Balaji et al., 2018). This makes this objective less natural for studying the hardness of look-ahead than the discounted or average objectives we consider, for which planning without look-ahead can be solved in polynomial time. Reinforcement learning with look-ahead is also studied in Merlis et al. (2024) for general $\ell \in \mathbb{N}^*$. However, the authors study *reward* look-ahead, whereas we study *transition* look-ahead. Furthermore, authors focus on value improvements while we study the computational complexity of planning.

A related line of work comes from the control literature, in particular Model Predictive Control (MPC) (Camacho and Bordons, 2013). MPC addresses the difficulty of accurately forecasting long-term system trajectories—especially under model misspecification or nonlinear dynamics—by repeatedly solving a simplified short-horizon optimization problem and then updating the control action according to the realized system state. In this sense, short-horizon system forecasts play a role analogous to look-ahead information. However, an important distinction is that in MPC, the forecasts are obtained by simulating an approximate model of the system, which may be misspecified and therefore not coincide with the true dynamics. By contrast, in our setting, transition look-ahead provides exact information about the actual next states under the environment's true transition kernel. Connections between MPC and reinforcement learning have been drawn in several works (Tamar et al., 2017; Efroni et al., 2020), primarily with the goal of improving planning efficiency or coping with disturbances. Some recent studies even analyze the dynamic regret or competitive ratio of controllers with partial look-ahead compared to those with full information (Li et al., 2019; Zhang et al., 2021; Lin et al., 2021, 2022). However, while prior studies on MPC-with-predictions analyze how well a controller performs given short-horizon forecasts, we ask a different question: how hard is it to compute an optimal plan when the planner has transition look-ahead in a discrete stationary MDP? In particular, our results prove that the tractable MPC controllers are necessarily sub-optimal.

Our work also connects to the broader literature on the computational complexity of solving MDPs. The foundational study of Papadimitriou (1987) establishes, among other results, that computing optimal policies in stationary MDPs is P-complete for both discounted and average objectives. Subsequent work investigated finite-horizon MDPs in greater detail (Mundhenk et al., 2000; Littman et al., 2013; Balaji et al., 2018), with Balaji et al. (2018) showing EXPTIME-hardness for planning in stationary MDPs under finite-horizon objectives. More recent results also address discounted MDPs (Chen and Wang, 2017).

Beyond these classical formulations, several works highlight how modifications of the *information structure* affect computational hardness. On the one hand, reducing information typically makes planning harder: for example, reinforcement learning with delayed feedback (where the agent receives rewards and/or transitions only after a lag) has been studied by Walsh et al. (2009), who show that planning in constant-delayed MDPs is NP-hard in general due to the exponential blowup of the augmented state space, while also proposing tractable algorithms for deterministic or mildly stochastic cases. More generally, POMDPs illustrate how partial observability raises the complexity to PSPACE-completeness in finite horizon and even undecidability in infinite horizon (Papadimitriou, 1987). On the other hand, our results show that *increasing* the information available to the agent—by granting exact transition look-ahead—can also lead to intractability: while one-step look-ahead remains efficiently solvable, multi-step look-ahead renders the planning problem NP-hard. Thus, transition look-ahead complements the existing literature by identifying a new axis where computational complexity undergoes a phase transition: both information loss and information gain can fundamentally alter the hardness of planning.

On the algorithmic side, linear–programming (LP) formulations (Puterman, 2014) for both discounted and average–reward MDPs have been established since the foundational works of Manne (1960); d'Epenoux (1963). This line of work advocates LP methods as an alternative to dynamic–programming approaches. The distinction matters here: Value Iteration (VI) and standard Policy Iteration (PI), although widely used and efficient in practice, do not admit polynomial–time guarantees neither in the discounted case (Feinberg

and Huang, 2013; Hollanders et al., 2012) nor in the average case (Fearnley, 2010). For one–step look-ahead ($\ell = 1$), our positive result gives an LP formulation that yields a polynomial-time algorithm. In sharp contrast, for $\ell \geq 2$ we prove NP–hardness, pinpointing the look-ahead depth as the tractability/intractability threshold in tabular MDPs.

Our setting is also related to the growing literature on algorithms with predictions (Mitzenmacher and Vassilvitskii, 2020; Benomar et al., 2025; Benomar and Perchet, 2025; Merlis et al., 2023). These works study how providing side information can help algorithms go beyond worst-case performance, often quantifying trade-offs between consistency (when predictions are accurate) and robustness (when predictions are wrong). In reinforcement learning, this perspective has recently been explored in several directions. Li et al. (2024) design a learning-augmented controller for LQR with latent perturbations, showing that accurate predictions yield near-optimality while robustness can still be preserved under prediction errors. Lyu et al. (2025) propose a framework where predictions on the transition matrix of discounted MDPs can reduce sample complexity bounds. Finally, Li et al. (2023) establish a consistency–robustness tradeoff when predictions come as Q-values in non-stationary MDPs. A key distinction from our contribution is that, while these works show that predictions can improve performance beyond worst-case guarantees, we show in the context of transition look-ahead that optimally leveraging such predictions can dramatically increase the computational complexity.

2 PROBLEM FORMULATION

2.1 Markov decision processes (MDP) and evaluation criteria

We study finite tabular Markov decision processes (MDPs)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r),$$

where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, $P(s' \mid s, a)$ denotes the probability of reaching state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, and $r : \mathcal{S} \times \mathcal{A} \to [0, R_{\max}]$ is the reward function.
A (possibly randomized) stationary memoryless policy[1] is a mapping

$$\pi : \mathcal{S} \to \Delta(\mathcal{A}),$$

where $\Delta(\mathcal{A})$ denotes the simplex over $\mathcal{A}$. Under such a policy, the interaction evolves as follows: at each time $t \in \mathbb{N}$, the system is in state $s_t \in \mathcal{S}$, the agent selects an action $a_t \sim \pi(\cdot \mid s_t)$, receives reward $r(s_t, a_t)$, and the next state is sampled according to

$$s_{t+1} \sim P(\cdot \mid s_t, a_t).$$

In this paper, we focus on the discounted return and the long-run average reward, which are the canonical objectives for which planning in standard MDPs is known to be polynomial-time solvable. By contrast, note that finite–horizon objectives are less suited to our complexity analysis as there is no polynomial planning algorithm in this case, even without look-ahead information (see Balaji et al. 2018).

Discounted return. The most classical formulation assigns geometrically decaying weights to future rewards (Puterman, 2014, Chapter 6). For a discount factor $\gamma \in (0, 1)$ and an initial state $s \in \mathcal{S}$, the value of a policy π is

$$v_\gamma^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s \right]. \tag{1}$$

The optimal discounted value function is defined by

$$v_\gamma^*(s) = \sup_\pi v_\gamma^\pi(s), \qquad \forall s \in \mathcal{S}, \tag{2}$$

where the supremum is taken over stationary memoryless policies. It is well known that v_γ^* is the unique solution of the Bellman optimality equations $\forall s \in \mathcal{S}$,

$$v_\gamma^*(s) = \max_{a \in \mathcal{A}} \Big\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a)\, v_\gamma^*(s') \Big\}. \tag{3}$$

and that the optimum is attained by a stationary deterministic policy.

Average reward criterion. A second perspective focuses on the asymptotic regime, where transient effects vanish and performance is measured by the long-run average reward (Puterman, 2014, Chapter 8). For a stationary policy π, the value for starting in s is

$$g^\pi(s) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[\sum_{t=0}^{T-1} r(s_t, a_t) \,\middle|\, s_0 = s,\ a_t \sim \pi(s_t) \right].$$

In the average case, it is standard to work under the unichain assumption:

Assumption 1 (Unichain MDP)**.** *An MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r)$, satisfies the Unichain assumption if for any stationary policy π, for any $s \in S$, the return time*

$$\tau_s \;=\; \inf\{t \geq 1 : S_t = s \mid S_0 = s\}$$

satisfies $\mathbb{E}_{P,\pi}[\tau_s] < \infty$.

[1] For both the discounted and average-reward criteria considered here, one can restrict without loss of optimality to stationary memoryless policies; see, e.g., Puterman (2014).

Assumption 1 ensures that $g^\pi(s)$ does not depend on s, so we simply write g^π. The transient deviations from this average are measured by the *bias function* $h^\pi : \mathcal{S} \to \mathbb{R}$, defined as

$$h^\pi(s) \;=\; \mathbb{E}^\pi\left[\sum_{t=0}^{+\infty}\big(r(s_t,a_t)-g^\pi\big)\;\middle|\; s_0=s\right],$$

and the optimal gain g^* is defined as

$$g^* \;=\; \sup_\pi\; g^\pi.$$

The optimal gain/bias pair (g^*, h^*) is uniquely characterized (for h^*, up to an additive constant) by the Bellman optimality equation:

$$\forall s \in \mathcal{S},$$
$$g^* + h^*(s) = \max_{a\in\mathcal{A}}\Big\{r(s,a) + \sum_{s'} P(s' \mid s,a)\, h^*(s')\Big\}.$$

2.2 Transition look-ahead as state augmentation

In the next section, we formalize the extra information provided by the look-ahead in terms of state observability and provide an augmented MDP construction that allows us to embed this problem into the standard evaluation framework introduced above.

2.2.1 Look-ahead and state observability

In the standard model, the agent only observes its current state s_t before acting. We extend this by allowing the agent to query an ℓ-step *transition look-ahead*: before choosing an action, the agent is provided with the entire ℓ-step transition tree rooted at s_t, i.e., all *realizations of future states* that may occur within ℓ steps under every possible action sequence.

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r)$ be the MDP in which the agent is provided with ℓ-step look-ahead information. To formalize this notion, we define the following random operator.

Definition 1 (Push-Forward Operator)**.** *Fix $t \in \mathbb{N}$ and an action sequence $\bar{a}_k = (a_1, \dots, a_k) \in \mathcal{A}^k$. The Push-Forward Operator, denoted by $\Pi[s_t, k, \bar{a}_k]$, is a random variable taking values in $\mathcal{S}$, corresponding to the state reached from s_t after k steps when actions $\bar{a}_k$ are applied.*

In other words, for any action sequence $\bar{a}_k$, the random variable $\Pi[s_t, k, \bar{a}_k]$ represents the state s_{t+k} that would be obtained if the sequence $\bar{a}_k$ were followed from time t.
The look-ahead information contains all realizations of ℓ-step trajectories and is formally defined as follows:

Definition 2 (ℓ-step transition look-ahead)**.** *An agent interacting with $\mathcal{M}$ is said to be provided with ℓ-step transition look-ahead if, for all $t \in \mathbb{N}$, it observes*

$$(\Pi[s_t, k, \bar{a}_k])_{\bar{a}_k\in\mathcal{A}^k,\; 0\le k\le \ell} \;\in\; \prod_{k=0}^{\ell} \mathcal{S}^{\mathcal{A}^k}.$$

before acting.

Remark 1. ***(i) Base case.*** *For all $t \in \mathbb{N}$,*

$$\Pi[s_t, 0, (\varnothing)] = s_t.$$

In particular, 0-step transition look-ahead corresponds to the standard observation without any transition look-ahead.
(ii) Recursion. *The collection of random variables $(\Pi[s_t, k, \bar{a}_k])$ is assumed to be* prefix-consistent*: for any $k \ge 1$ and any $\bar{a}_{k+1} \in \mathcal{A}^{k+1}$,*

$$\Pi[\Pi[s_t, 1, a_1], k, \bar{a}_{2:k+1}] = \Pi[s_t, k+1, \bar{a}_{k+1}]. \tag{4}$$

(iii) Distribution. *For any $s \in \mathcal{S}$ and any $\bar{a}_k = (a_1, \dots, a_k)$,*

$$\begin{aligned}&\mathbb{P}\big(\Pi[s_t, k, \bar{a}_{1:k}] = s'\big)\\ &= \mathbb{E}_{\Pi[s_t,k-1,\bar{a}_{1:k-1}]}\big[P\big(s' \mid \Pi[s_t, k-1, \bar{a}_{1:k-1}], a_k\big)\big]\,.\end{aligned} \tag{5}$$

2.2.2 Augmented MDP

State and action space We construct an *augmented MDP* $\bar{\mathcal{M}} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}, \bar{r})$ such that an agent interacting with $\mathcal{M}$ with ℓ-step transition look-ahead is equivalent to a standard agent interacting in $\bar{\mathcal{M}}$.
Let $t \in \mathbb{N}$. We define the *augmented state* $\xi_t \in \bar{\mathcal{S}}$ as

$$\bar{\mathcal{S}} = \prod_{k=0}^{\ell} \mathcal{S}^{\mathcal{A}^k},$$

and

$$\xi_t = (\xi_t[k])_{0\le k\le \ell}\,, \qquad \text{where } \forall k \in \{0, \dots, \ell\},$$
$$\xi_t[k] = (\Pi[s_t, k, \bar{a}_k])_{\bar{a}_k\in\mathcal{A}^k} \in \mathcal{S}^{\mathcal{A}^k}.$$

Note that although the look-ahead reveals the outcomes of *hypothetical* action trajectories of length ℓ, the agent has no incentive to commit in advance to executing the entire sequence. Indeed, if at time t the agent fixes an action sequence $(a_t, \dots, a_{t+\ell-1})$, then the last $\ell - 1$ actions are chosen without exploiting the additional look-ahead information that will be revealed at subsequent steps. Such a strategy therefore uses strictly less information than a policy that selects actions sequentially, conditioning on newly observed look-ahead signals. Consequently, it is without loss of optimality to restrict attention to policies that select only the current action.
The action set is therefore unchanged:

$$\bar{\mathcal{A}} = \mathcal{A}.$$

Transition and reward model For any $j, k, l \in \mathbb{N}$ with $j < k < l$ and any sequence $\bar{a} \in \mathcal{A}^l$, we denote $\bar{a}_{j:k} = (a_j, \ldots, a_k)$ and $\bar{a}_j = \bar{a}_{1:j}$. We also define, for $i \leq \ell$ and any prefix $\bar{a}'_j \in \mathcal{A}^j$,

$$\xi_t[i](\bar{a}'_j) = (\Pi[s_t, i, \bar{a}_i])_{\substack{\bar{a}_i \in \mathcal{A}^i \\ \bar{a}_j = \bar{a}'_j}},$$

i.e., the restriction of the i-step look-ahead to action sequences with prefix $\bar{a}'_j$.

For any $t \in \mathbb{N}$, the first ℓ blocks of ξ_t evolve deterministically given (ξ_t, a_t), due to the consistency property of Π (see (4)). More precisely, for $k = 0, \ldots, \ell-1$,

$$\xi_{t+1}[k] = \xi_t[k+1](a_t),$$

where $\xi_t[k+1](a_t)$ denotes the restriction of $\xi_t[k+1]$ to action sequences whose first action is a_t. By contrast, the last block $\xi_{t+1}[\ell]$ is stochastic, as it corresponds to newly generated look-ahead values.

The transition kernel $\bar{P}$ is therefore defined as follows: for any $\xi, \xi' \in \bar{\mathcal{S}}$ and $a \in \mathcal{A}$,

$$\begin{aligned}
&\bar{P}_a(\xi, \xi') = \\
&\mathbb{1}\{\forall k = 0, \ldots, \ell-1,\ \xi'[k] = \xi[k+1](a)\} \\
&\quad \times \mathbb{P}\Big(\xi'[\ell] = \big(\Pi[s_{t+1}, \ell, \bar{a}_\ell]\big)_{\bar{a}_\ell \in \mathcal{A}^\ell} \,\Big|\, \xi_t = \xi,\ a_t = a\Big).
\end{aligned}$$

We refer the reader to Appendix S1 for detailed computation of the transition kernel.

Finally, let $\xi \in \bar{\mathcal{S}}, a \in \mathcal{A}$ the reward function in the augmented MDP is defined by

$$\bar{r}(\xi, a) = r(\xi[0], a),$$

since $\xi[0]$ encodes the current state.

2.3 Decision problems

To analyze the computational complexity of planning with transition look-ahead, we work with standard *decision-problem* formulations. These are classical complexity–theoretic encodings of the main planning objectives (discounted and average reward).

We focus on ℓ-look-ahead decision problems where the agent is endowed with ℓ–step transition look-ahead as defined in the previous section. The look-ahead depth $\ell \in \mathbb{N}$ is thus a fixed constant of the problem definition and not an input parameter. In particular, this implies that an algorithm that solves the decision problem after say $(\mathcal{S}\mathcal{A})^\ell$ operations is polynomial. Our complexity results should be interpreted in the same spirit as classical k–SAT: while 2–SAT is polynomially solvable, 3–SAT is NP-hard. Analogously, we establish that planning is tractable for $\ell = 1$, but NP-hard for $\ell \geq 2$.

Formally, each problem takes as input an MDP instance together with parameters describing the evaluation criterion, and asks whether there exists a (possibly randomized) policy whose value exceeds a prescribed threshold. Following the standards in complexity theory, the numerical values of the input, such as the discount factor γ, rewards, transition matrix, or threshold θ, are encoded in binary. It implies that an algorithm with $O(\log(R_{max}))$ complexity is polynomial (where R_{max} is the maximum of the reward function), but not an algorithm with complexity $O(R_{max})$.

Definition 3 (Discounted Value Decision Problem (ℓ-DVDP))**.** *Instance: a finite MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r)$, an initial state $s_0 \in \mathcal{S}$, $\gamma \in (0, 1)$, and $\theta \in \mathbb{R}$.*
Question: Does there exist a policy (possibly randomized) π such that

$$v^\pi_\gamma(s_0; \mathcal{M}, \ell) \ \geq\ \theta\ ? \tag{6}$$

Definition 4 (Average-Reward Decision Problem (ℓ-ARDP))**.** *Instance: a finite MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r)$ and $\theta \in \mathbb{R}$.*
Question: Does there exist a stationary (possibly randomized) policy π such that

$$g^\pi(\mathcal{M}, \ell) \ \geq\ \theta\ ? \tag{7}$$

These decision problems will serve as our canonical complexity-theoretic objects. When $\ell = 0$ (no look-ahead), they are solvable in polynomial time via classical LP formulations; we will show that tractability extends to $\ell = 1$, while $\ell \geq 2$ renders each problem NP-hard, delineating a sharp complexity frontier for transition look-ahead.

We emphasize that these decision problems are the right vehicle for hardness. If there exists an algorithm that can compute the optimal value in polynomial time, it can be used to answer the decision question in one call by comparing its optimal value to θ. Therefore, the hardness of the decision problem implies the hardness of finding the optimal value function. In the other direction, an oracle that can solve the decision problems in polynomial time can be used to compute the optimal value up to any desired accuracy $\varepsilon > 0$ by bisection on θ with a complexity polynomial in the size of the input and $\log(1/\varepsilon)$. In this study, however, we solve the decision problems in the case $\ell = 1$ by computing the optimal value and the optimal policy *exactly*.

3 PLANNING WITH ONE STEP TRANSITION LOOK-AHEAD

When $\ell = 1$, the look-ahead representation simplifies to observing, at time t before acting in s_t, the collection of one-step successors $\{\Pi[s_t, 1, a]\}_{a \in \mathcal{A}} \in \mathcal{S}$. Note

that in this case, $\xi \in \bar{\mathcal{S}}$ reduces to (s, p), where $s \in \mathcal{S}$ and $p \in \mathcal{S}^{\mathcal{A}}$. Therefore, the agent chooses its action after observing the entire vector of next states. In the following, we focus on planning in this setting. For clarity of exposition, we restrict attention to the discounted criterion; the arguments extend with only minor changes to the average–reward case, as will be explained at the end of this section and in more detail in Appendices S2.1 and S2.2.

A natural way to characterize one-step look-ahead planning is to write the linear program directly in the augmented state space. In the discounted case, for $\gamma \in (0, 1)$ and strictly positive weights $(\bar{\mu}(\xi))_{\xi \in \bar{\mathcal{S}}}$, the optimal value function $\bar{v}^* : \bar{\mathcal{S}} \to \mathbb{R}$ can be obtained as the solution of:

$$\min_{\bar{v}} \sum_{\xi \in \bar{\mathcal{S}}} \bar{\mu}(\xi)\, \bar{v}(\xi), \quad \text{s.t. } \forall \xi \in \bar{\mathcal{S}} :$$
$$\bar{v}(\xi) \;\geq\; \max_{a \in \mathcal{A}} \Big\{ \bar{r}(\xi, a) + \gamma\, \mathbb{E}_{\xi' \sim \bar{P}_a(\xi)} \big[\bar{v}(\xi') \big] \Big\}. \tag{8}$$

Although (8) is written over the augmented state space and thus involves a value function $\bar{v}$ indexed by exponentially many augmented states, it admits an equivalent polynomial-size reformulation, which ensures tractability in the discounted setting.
Theorem 1 shows that in finite tabular MDPs, planning with one-step transition look-ahead is solvable in polynomial time for both (i) the discounted and (ii) the average-reward criteria.

Theorem 1 (One-step look-ahead is polynomial–time)**.** *ℓ-DVDP and ℓ-ARDP are solvable in polynomial time for $\ell \leq 1$.*

Note that the proof is constructive: we explicitly encode the planning problem as a linear program whose feasible region captures the one-step look-ahead dynamics. Solving this LP yields the optimal value and an associated optimal policy in polynomial time.

Proof sketch (discounted case). We consider the discounted objective with transition look-ahead of depth $\ell = 1$, where at time t the agent observes the entire next-state vector

$$p \in \mathcal{S}^{\mathcal{A}}$$

before selecting an action. In this case, an augmented state is a pair $(s, p) \in \mathcal{S} \times \mathcal{S}^{\mathcal{A}}$, where s is the current state and $p(a)$ is the next state that would be reached if action a were played. For each $s \in \mathcal{S}$, define the distribution

$$Q(\cdot \mid s) \in \Delta(\mathcal{S}^{\mathcal{A}})$$

by

$$Q(p \mid s) = \prod_{a \in \mathcal{A}} P(p(a) \mid s, a), \qquad p \in \mathcal{S}^{\mathcal{A}}.$$

Thus, conditional on the current state s, the one-step look-ahead vector p is obtained by drawing independently one successor for each action.
We show Lemma S1 that the optimal value function $\bar{v}^*$ of the augmented MDP can be expressed in terms of a value function $v^* : \mathcal{S} \to \mathbb{R}$ as

$$\bar{v}^*(s, p) = \max_{a \in \mathcal{A}} \{ r(s, a) + \gamma v^*(p(a)) \}, \;\; \forall (s, p) \in \mathcal{S} \times \mathcal{S}^{\mathcal{A}},$$

where v^* is the unique solution of

$$v(s) = \mathbb{E}_{p \sim Q(\cdot|s)} \left[\max_{a \in \mathcal{A}} \{ r(s, a) + \gamma v(p(a)) \} \right], \;\; \forall s \in \mathcal{S}. \tag{9}$$

Moreover, for any strictly positive measure μ on $\mathcal{S}$, this fixed-point equation is equivalently characterized by the linear program

$$\min_{v : \mathcal{S} \to \mathbb{R}} \quad (1 - \gamma) \sum_{s \in \mathcal{S}} \mu(s)\, v(s) \quad \text{s.t. } \forall s \in \mathcal{S} \tag{10}$$
$$v(s) \geq \mathbb{E}_{p \sim Q(\cdot|s)} \left[\max_{a \in \mathcal{A}} \{ r(s, a) + \gamma v(p(a)) \} \right], \tag{11}$$

The remaining difficulty is that the maximization operator appears inside the expectation in (11). Expanding the expectation over all $p \in \mathcal{S}^{\mathcal{A}}$ would lead to exponentially many terms. To handle this, we use the ellipsoid method together with a polynomial-time separation oracle.
Fix $s \in \mathcal{S}$, let

$$u_{v,s}(s', a) = r(s, a) + \gamma v(s'), \qquad (s', a) \in \mathcal{S} \times \mathcal{A},$$

and let $N = |\mathcal{S}||\mathcal{A}|$. Index the pairs (s', a) as $\{(s^i, a^i)\}_{i=1}^N$. For any permutation m of $\{1, \ldots, N\}$, define the events

$$E_i^m = \Big\{ p \in \mathcal{S}^{\mathcal{A}} : \; i = \min\big\{ j \in [N] : \, p(a^{m(j)}) = s^{m(j)} \big\} \Big\}.$$

These events form a partition of $\mathcal{S}^{\mathcal{A}}$. Therefore,

$$\mathbb{E}_{p \sim Q(\cdot|s)} \left[\max_{a \in \mathcal{A}} u_{v,s}(p(a), a) \right] = \sum_{i=1}^{N} Q(E_i^m \mid s)\, u_{v,s}(s^{m(i)}, a^{m(i)})$$

whenever m orders the pairs (s', a) in decreasing order of $u_{v,s}(s', a)$. This yields an equivalent family of linear constraints, $\forall s \in \mathcal{S}, \; \forall m \in \mathcal{L}$,

$$v(s) \geq \sum_{i=1}^{N} Q(E_i^m \mid s) \big(r(s, a^{m(i)}) + \gamma v(s^{m(i)}) \big), \tag{12}$$

where $\mathcal{L}$ denotes the set of all permutations of $\{1, \ldots, N\}$.
We then show in Lemma S2 that (12) admits a polynomial-time separation oracle. Given a candidate v, for each state s it suffices to sort the pairs (s', a) in decreasing order of $u_{v,s}(s', a)$, obtaining a permutation $m_{u_{v,s}}$, and to check only the corresponding constraint. The probabilities $Q(E_i^{m_{u_{v,s}}} \mid s)$ can be computed in polynomial time using the product structure of $Q(\cdot \mid s)$.
This yields a polynomial-time separation oracle for (10)–(11). By the ellipsoid method, the reduced LP can therefore be solved in polynomial time. □

The average-reward case follows by the same approach; full details are given in Appendix S2.2.

4 PLANNING WITH TWO OR MORE STEPS OF TRANSITION LOOK-AHEAD

We now show that allowing look-ahead of horizon $\ell \geq 2$ fundamentally changes the computational nature of planning.
Theorem 2 shows that for finite tabular MDPs, the ℓ-step transition look-ahead planning problem is NP-hard for any $\ell \geq 2$ in the discounted setting.

Theorem 2 (NP-hardness for $\ell \geq 2$ (discounted))**.** *For any $\ell \geq 2$, ℓ-DVDP is NP-hard.*

The discounted case serves as the cornerstone of our argument. We establish NP-hardness for $\ell = 2$, which extends to all larger look-ahead horizons.

Proof. Given random variables $X_1, \ldots, X_n$, an integer k and a threshold C, the Largest Expected Value problem consists in deciding whether

$$\max_{Y \subset [n] : |Y| = k} \mathbb{E}[\max_{i \in Y} X_i] \geq C. \tag{13}$$

Largest Expected Value has been shown to be NP-hard (Mehta et al., 2020). The key idea is to connect to this problem by constructing an MDP instance where computing the optimal policy requires solving a closely related subset-selection problem. More precisely, our proof relies on the gap construction of Mehta et al. (2020), which reduces Independent Set on regular graphs to such a stochastic optimization problem.

Let $G = (V, E)$ be an undirected 3-regular graph. Using the random variables $(X_v)_{v \in V}$ provided by the reduction of Mehta et al. (2020), we construct an MDP $\mathcal{M}_G = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ whose structure is summarized in Figure 1 and described in detail in Appendix S3. The state space contains a root state s_0, a selector state s_1, vertex states $(s_v)_{v \in V}$, support states encoding the realizations of the random variables $(X_v)_{v \in V}$, and an absorbing terminal state s_T. From s_0, action wait loops back to s_0, while action go moves deterministically to s_1. From s_1, the actions $\mathsf{pick}_1, \ldots, \mathsf{pick}_k$ reveal candidate vertex states, and from a vertex state s_v, action claim samples the corresponding random payoff X_v. Rewards are only collected on the support states before reaching s_T.

With $\ell = 2$ look-ahead, an agent at s_0 observes the candidate vertices that would become available after committing through go and then choosing one of the selector actions. More precisely, each augmented root state $\xi = (s_0, p_1, p_2)$ reveals a k-tuple of candidate vertices, which induces a subset $S_V(\xi) \subseteq V$. If the agent plays wait, it remains in s_0 and resamples the two-step look-ahead, thereby drawing a fresh candidate set. If it instead commits through go, it transitions to s_1 and then chooses among the currently revealed candidates. Let X_v denote the random payoff obtained by claiming at vertex state s_v. The key structural result (Lemma S5) shows that for every augmented root state ξ,

$$\bar{v}^\star(\xi) = \max \left\{ \gamma^3 \, \mathbb{E}\left[\max_{v \in S_V(\xi)} X_v \right], \; \gamma \, \mathbb{E}_{\xi' \sim \bar{P}_{\mathsf{wait}}(\cdot \mid \xi)}[\bar{v}^\star(\xi')] \right\}.$$

Thus, at the root, the agent trades off immediate commitment to the currently revealed subset against waiting for a better one.

In the NO case, every subset of size at most k has expected maximum at most $k\mu - 1$, which yields a uniform upper bound on the root value. In the YES case, there exists a subset $S^\star$ of size k such that waiting until the look-ahead reveals exactly $S^\star$ and then committing yields a strictly larger value. Choosing $\gamma < 1$ sufficiently close to 1 separates the two cases and therefore gives a polynomial-time reduction from Independent Set to 2-*DVDP*. This proves NP-hardness. □

We then show in Theorem 3 that hardness also extends to the average–reward criterion.

Theorem 3 (NP-hardness for $\ell \geq 2$ (average reward))**.** *For any $\ell \geq 2$, ℓ-ARDP is NP-hard.*

Proof sketch. We prove NP-hardness of the average–reward case by a reduction from the discounted setting (Theorem 2). Let $\bar{\mathcal{M}}_G$ be the augmented MDP used to prove Theorem 2. We modify $\bar{\mathcal{M}}_G$ by adding an independent Bernoulli "reset" coin at each step: with probability $1 - \gamma$, the process resets to a fresh root augmented state distributed according to the look-ahead law at s_0, and with probability γ it follows the original transition kernel $\bar{P}$. Rewards are left unchanged. Thus the dynamics of the modified MDP

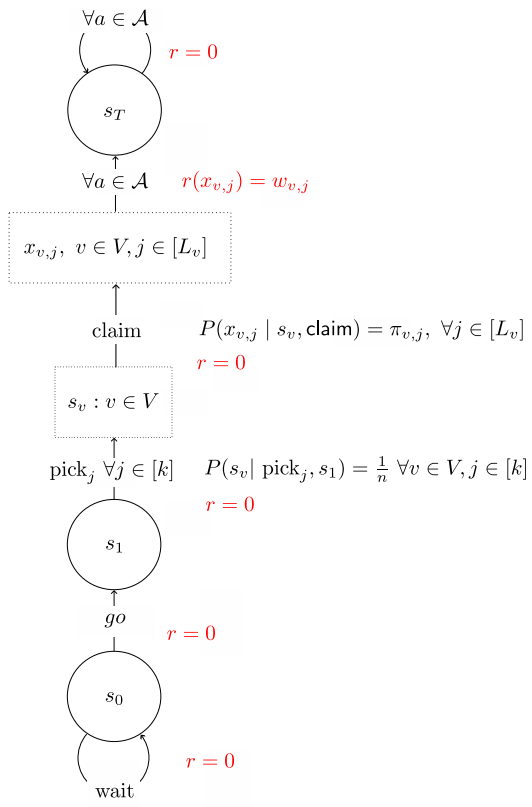

Figure 1: **Hardness of 2-*DVDP*.** At time $t = 0$, the agent is at the root state s_0 and observes, through depth-2 look-ahead, the candidate vertex states that would become available after playing go and then one of the selector actions $\mathsf{pick}_1, \ldots, \mathsf{pick}_k$. At s_0, it may either play wait to remain in s_0 and obtain a fresh look-ahead, or play go to transition to s_1. Once in s_1, the agent chooses among the currently revealed candidates; if it then moves to a vertex state s_v, the subsequent payoff is governed by the random variable X_v. Thus, if the agent commits at time τ, the value of the revealed look-ahead is of the form $\mathbb{E}[\max_{v \in S_V(\xi_\tau)} X_v]$. Optimizing the time at which the agent commits at s_0 is then shown to be essentially as hard as the stochastic subset-selection problem underlying the reduction of Mehta et al. (2020).

$\bar{\mathcal{M}}'_G$ are:

$$\bar{P}'_a(\xi', \xi) = \gamma\, \bar{P}_a(\xi', \xi) + (1 - \gamma)\, \Lambda_{s_0}(\xi'),$$
$$\bar{r}'(\xi, a) = \bar{r}(\xi, a).$$

Because the reset coin is tossed independently at each step, the trajectory of $\bar{\mathcal{M}}'_G$ naturally decomposes into i.i.d. cycles between successive resets. A cycle has expected length $\frac{1}{1-\gamma}$, and the expected cumulative reward of a cycle under any stationary policy π coincides with the discounted return in $\bar{\mathcal{M}}_G$. By the Renewal–Reward Theorem, the long-run average reward in $\bar{\mathcal{M}}'_G$ satisfies

$$g^\pi(\bar{\mathcal{M}}'_G) \;=\; (1 - \gamma)\, \bar{v}^\pi_\gamma(\xi_0; \bar{\mathcal{M}}_G).$$

Therefore, given a threshold θ in the discounted instance, we define

$$\kappa \triangleq (1 - \gamma)\theta.$$

Then

$$\bar{v}^\pi_\gamma(\xi_0; \bar{\mathcal{M}}_G) \geq \theta \iff g^\pi(\bar{\mathcal{M}}'_G) \geq \kappa.$$

Hence, deciding whether there exists a policy exceeding θ in the discounted setting is equivalent to deciding whether there exists a policy exceeding κ in the average–reward setting. The detailed proof is provided in Appendix S4. □

5 CONCLUSION AND FUTURE WORK

This work identifies a sharp computational frontier for planning with transition look-ahead. By introducing canonical decision formulations for both the discounted and average-reward criteria, we establish that planning is tractable in polynomial time for one-step look-ahead ($\ell = 1$), with explicit linear programming formulations (Theorem 1), whereas for $\ell \geq 2$ the problem becomes NP-hard under both criteria (Theorems 2 and 3). This dichotomy mirrors classical complexity thresholds such as the jump from 2–SAT to 3–SAT, and shows that while deeper look-ahead enriches the agent's information, it simultaneously induces a combinatorial explosion that makes exact planning computationally intractable.

Note that our NP-hardness results do not imply that planning with $\ell \geq 2$ is unsolvable, but rather that exact solutions cannot be expected in full generality. This motivates several directions for future work. On the approximation side, it remains open whether polynomial-time approximation schemes (PTAS) exist for discounted ℓ–look-ahead planning, or conversely, whether even constant-factor approximation is impossible. On the structural side, one may ask under which

restrictions hardness disappears: our reduction relies on constructing a worst-case instance that crucially uses dense and irregular transition structures. Hence, tractability may be recovered when the MDP satisfies additional structure, such as factored dynamics, sparse transition graphs, or monotone rewards (i.e., rewards that are non-decreasing along the natural partial order induced by the state space, which precludes the oscillatory patterns needed by our reduction). Similarly, when the discount factor γ is sufficiently small—for instance, $\gamma < 0.5$, which dampens long-term dependencies—the reduction no longer applies, suggesting that hardness may vanish. Another direction is to study the complexity of near-optimal solutions under restricted policy classes, e.g., policies constrained by structural priors such as monotonicity or threshold rules. Beyond exact look-ahead, a natural extension is to allow noisy or costly predictions, and to analyze how robustness and budget constraints interact with hardness. Our results also bear on learning: they confirm that model-predictive-control (MPC) style strategies—which rely on short-horizon roll-outs and commit to open-loop action sequences—are inherently suboptimal in the tabular case. This raises a broader algorithmic question: how can one design learning procedures that remain computationally efficient while approximating the (generally intractable) optimal planner with deeper look-ahead?

Acknowledgments

This research was supported by the Israel Science Foundation (ISF, Grant No. 4118/25) and the Maimonides Fund's Future Scientists Center. The authors thank Simon Mauras for insightful discussions notably on LP reduction. We also thank the anonymous reviewers for their valuable feedback and constructive comments.

References

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International conference on machine learning*, pages 263–272. PMLR, 2017.

Nikhil Balaji, Stefan Kiefer, Petr Novotnỳ, Guillermo A Pérez, and Mahsa Shirmohammadi. On the complexity of value iteration. *arXiv preprint arXiv:1807.04920*, 2018.

Ziyad Benomar and Vianney Perchet. On tradeoffs in learning-augmented algorithms, 2025. URL https://arxiv.org/abs/2501.12770.

Ziyad Benomar, Lorenzo Croissant, Vianney Perchet, and Spyros Angelopoulos. Pareto-optimality, smoothness, and stochasticity in learning-augmented one-max-search, 2025. URL https://arxiv.org/abs/2502.05720.

Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2 edition, 2013. ISBN 978-0-85729-398-5. doi: 10.1007/978-0-85729-398-5.

Yichen Chen and Mengdi Wang. Lower bound on the computational complexity of discounted markov decision problems, 2017. URL https://arxiv.org/abs/1705.07312.

F. d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, 1963. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2627210.

Yonathan Efroni, Mohammad Ghavamzadeh, and Shie Mannor. Online planning with lookahead policies, 2020. URL https://arxiv.org/abs/1909.04236.

John Fearnley. Exponential lower bounds for policy iteration, 2010. URL https://arxiv.org/abs/1003.3418.

Eugene A. Feinberg and Jefferson Huang. The value iteration algorithm is not strongly polynomial for discounted dynamic programming, 2013. URL https://arxiv.org/abs/1312.6832.

Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2): 169–197, 1981. doi: 10.1007/BF02579273.

Romain Hollanders, Jean-Charles Delvenne, and Raphaël M. Jungers. The complexity of policy iteration is exponential for discounted markov decision processes. In *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, pages 5997–6002, Maui, HI, USA, December 2012. IEEE. doi: 10.1109/CDC.2012.6426485. URL https://perso.uclouvain.be/romain.hollanders/docs/CDC12_HollandersDelvenneJungers_final_letter.pdf.

Thomas Jaksch, Rudolf Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. In *Journal of Machine Learning Research*, volume 11, pages 1563–1600, 2010.

Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 4863–4873, 2018.

Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979. English translation: Soviet Math. Dokl. 20:191–194.

Tongxin Li, Yiheng Lin, Shaolei Ren, and Adam Wierman. Beyond black-box advice: Learning-augmented algorithms for mdps with q-value predictions, 2023. URL https://arxiv.org/abs/2307.10524.

Tongxin Li, Hao Liu, and Yisong Yue. Disentangling linear quadratic control with untrusted ml predictions. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 86860–86898. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9dff3b83d463fab213941bfee23341ba-Paper-Conference.pdf.

Yingying Li, Xin Chen, and Na Li. Online optimal control with linear dynamics and predictions: Algorithms and regret analysis, 2019. URL https://arxiv.org/abs/1906.11378.

Yiheng Lin, Yang Hu, Haoyuan Sun, Guanya Shi, Guannan Qu, and Adam Wierman. Perturbation-based regret analysis of predictive control in linear time varying systems, 2021. URL https://arxiv.org/abs/2106.10497.

Yiheng Lin, Yang Hu, Guannan Qu, Tongxin Li, and Adam Wierman. Bounded-regret mpc via perturbation analysis: Prediction error, constraints, and nonlinearity, 2022. URL https://arxiv.org/abs/2210.12312.

Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems, 2013. URL https://arxiv.org/abs/1302.4971.

Lixing Lyu, Jiashuo Jiang, and Wang Chi Cheung. Efficiently solving discounted mdps with predictions on transition matrices, 2025. URL https://arxiv.org/abs/2502.15345.

Alan S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2627340.

Aranyak Mehta, Uri Nadav, Alexandros Psomas, and Aviad Rubinstein. Hitting the high notes: Subset selection for maximizing expected order statistics, 2020. URL https://arxiv.org/abs/2012.07935.

Nadav Merlis. Reinforcement learning with lookahead information, 2024. URL https://arxiv.org/abs/2406.02258.

Nadav Merlis, Hugo Richard, Flore Sentenac, Corentin Odic, Mathieu Molina, and Vianney Perchet. On preemption and learning in stochastic scheduling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24478–24516. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/merlis23a.html.

Nadav Merlis, Dorian Baudry, and Vianney Perchet. The value of reward lookahead in reinforcement learning, 2024. URL https://arxiv.org/abs/2403.11637.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions, 2020. URL https://arxiv.org/abs/2006.09123.

Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon markov decision process problems. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, pages 494–499, 2000.

C. H. Papadimitriou. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Hoboken, NJ, 2nd edition, 2014.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

Aviv Tamar, Garrett Thomas, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Learning from the hindsight plan – episodic mpc improvement, 2017. URL https://arxiv.org/abs/1609.09001.

Shoshana Vasserman, Michal Feldman, and Avinatan Hassidim. Implementing the wisdom of waze. In *IJCAI*, volume 15, pages 660–666, 2015.

Thomas J. Walsh, Ali Nouri, Lihong Li, and Michael L. Littman. Learning and planning in environments with delayed feedback. *Autonomous Agents and Multi-Agent Systems*, 18(1):83–105, 2009. doi: 10.1007/s10458-008-9056-7.

Runyu Zhang, Yingying Li, and Na Li. On the regret analysis of online LQR control with predictions. *arXiv preprint*, 2021. doi: 10.48550/arXiv.2102.01309. Submitted on 2 February 2021.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]