

Exponential Hardness of Reinforcement Learning with Linear Function Approximation

Daniel Kane

University of California, San Diego

DAKANE@ENG.UCSD.EDU

Sihan Liu

University of California, San Diego

SIL046@UCSD.EDU

Shachar Lovett

University of California, San Diego

SLOVETT@CS.UCSD.EDU

Gaurav Mahajan

Yale University

GAURAV.MAHAJAN@YALE.EDU

Csaba Szepesvári

DeepMind, London, UK

University of Alberta, Edmonton, Canada

SZEPESVA@UALBERTA.CA

Gellért Weisz

DeepMind, London, UK

University College London, London, UK

GELLERT@DEEPMIND.COM

Editors: Gergely Neu and Lorenzo Rosasco

Abstract

A fundamental question in reinforcement learning theory is: suppose the optimal value functions are linear in given features, can we learn them efficiently? This problem’s counterpart in supervised learning, linear regression, can be solved both statistically and computationally efficiently. Therefore, it was quite surprising when a recent work [Kane et al. \(2022\)](#) showed a computational-statistical gap for linear reinforcement learning: even though there are polynomial sample-complexity algorithms, unless $\text{NP} = \text{RP}$, there are no polynomial time algorithms for this setting.

In this work, we build on their result to show a computational lower bound, which is exponential in feature dimension and horizon, for linear reinforcement learning under the Randomized Exponential Time Hypothesis. To prove this we build a round-based game where in each round the learner is searching for an unknown vector in a unit hypercube. The rewards in this game are chosen such that if the learner achieves large reward, then the learner’s actions can be used to simulate solving a variant of 3-SAT, where (a) each variable shows up in a bounded number of clauses (b) if an instance has no solutions then it also has no solutions that satisfy more than $(1-\epsilon)$ -fraction of clauses. We use standard reductions to show this 3-SAT variant is approximately as hard as 3-SAT. Finally, we also show a lower bound optimized for horizon dependence that almost matches the best known upper bound of $\exp(\sqrt{H})$.

1. Introduction

Efficiently exploring and planning in environments with large state spaces is a central problem in reinforcement learning. Recently, there has been a lot of success in applying function approximation to classical reinforcement learning algorithms leading to state-of-the-art results in various practical applications.

This has also led to a growing interest of the reinforcement learning (RL) theory community to design and analyze efficient algorithms for the large state space regime. In this regime, the goal is to design algorithms whose complexity does not polynomially depend on the size of the state space. Since, this is impossible when we do not make any assumptions about the environment, much effort has been spent on finding minimal assumptions under which an optimal policy can be found efficiently: State Aggregation (Li, 2009; Dong et al., 2020), Linear q^π (Du et al., 2020a; Lattimore et al., 2020; Yin et al., 2022; Weisz et al., 2022a), Linear MDPs (Yang and Wang, 2019; Jin et al., 2020), Linear Mixture MDPs (Modi et al., 2020; Ayoub et al., 2020; Zhou et al., 2021), Reactive POMDPs (Krishnamurthy et al., 2016), Block MDPs (Du et al., 2019), FLAMBE (Agarwal et al., 2020), Reactive PSRs (Littman et al., 2001), Linear Bellman Complete (Munos, 2005; Zanette et al., 2020), Bellman rank (Jiang et al., 2016), Witness rank (Sun et al., 2019), Bilinear Classes (Du et al., 2021), Bellman Eluder (Jin et al., 2021) and Decision-Estimation Coefficient (Foster et al., 2021).

One such minimal assumption that came out of this line of work is RL with linear function approximation: when the optimal value function (either Q^* , or V^* , or both) can be obtained as the linear combination of finitely many, known basis functions. When both the optimal value functions Q^* and V^* satisfy this assumption (called linear $Q^* \& V^*$ henceforth), there are two *sample efficient* algorithms in the literature whose sample complexities are polynomial in the number of basis functions d and horizon H . First, the algorithm by Du et al. (2021) additionally assumes that the basis functions' values can be known and pre-processed for the whole state-action space. Second, TensorPlan (Weisz et al., 2021b, 2022b) replaces this with an implicit assumption that the number of actions is a small constant (as its sample complexity is exponential in this number). Weisz et al. (2021a, 2022b) showed sample complexity lower bounds exponential in $\min(d, H)$ that imply statistical hardness of finding a near-optimal policy when the number of actions is polynomial in d and the values of basis functions are only revealed for the sampled states. This indicates that one of the two aforementioned additional assumptions are required for a sample efficient algorithm. However, even when both additional assumptions are met, these works leave finding a computationally efficient algorithm for this setting as an important open question.

A recent work (Kane et al., 2022) made progress on this question by showing a computational-statistical gap in RL with linear function approximation: unless $\text{NP}=\text{RP}$, there is no polynomial time algorithm even for the easiest setting of linear $Q^* \& V^*$, deterministic transition, stochastic rewards and 2 actions. This is surprising because if we also assume that the rewards are deterministic, then this problem can be solved in $O(dH)$ time (Wen and Van Roy, 2017). Therefore, the result of (Kane et al., 2022) showed that adding noise in rewards can lead to computational intractability (similar transition happens for sample complexity if the number of actions is unrestricted Weisz et al. (2021a, 2022b)). However, the lower bound of (Kane et al., 2022) is not tight: they showed a quasi-polynomial lower bound in d whereas the best known upper bounds are exponential in $\min(d, H)$ (Du et al., 2020b).

2. Our Contributions

In this work, we provide almost matching exponential computational lower bounds for RL with linear function approximation. Before stating our main results, we first need to state some key definitions that we use throughout the paper.

2.1. Preliminaries

Markov Decision Process (MDP). We begin by defining the framework for reinforcement learning, a Markov Decision Process (MDP). We define a deterministic transition MDP as a tuple $M = (\mathcal{S}, \mathcal{A}, R, P)$, where \mathcal{S} is the state space, \mathcal{A} is the action space, $R : \mathcal{S} \times \mathcal{A} \mapsto \Delta([0, 1])$ is the stochastic reward function,¹ and $P : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the deterministic transition function. Such an MDP M gives rise to a discrete time sequential decision process where an agent starts from a starting state $S_0 \in \mathcal{S}$. Then, at each time t , the agent at some current state S_t , takes action A_t , receiving reward $R_t \sim R(S_t, A_t)$ and transitions to next state $S_{t+1} = P(S_t, A_t)$. This goes on until the agent reaches the end state $\perp \in \mathcal{S}$. In H -horizon problems each such trajectory/path from the starting state s_0 to an end state \perp is of length of at most H , and the sets of states \mathcal{S}_t that are reachable after t steps (taking any actions) are disjoint for $0 \leq t \leq H$. The goal of the decision making agent is to maximize the sum of the total expected rewards it receives along such a trajectory. As it turns out, the total expected reward regardless the initial state is achievable by following a deterministic, stationary policy, which is given by some map $\pi : \mathcal{S} \mapsto \mathcal{A}$ and following π means that in step t if the state is S_t , the action taken is $A_t = \pi(S_t)$. Given a policy π and a state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, we let

$$V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^{\tau-1} R(S_t, A_t) \mid S_0 = s, \pi \right], \quad Q^\pi(s, a) = \mathbb{E} \left[\sum_{t=0}^{\tau-1} R(S_t, A_t) \mid S_0 = s, A_0 = a, \pi \right]$$

denote the total expected reward where $S_1, A_1, \dots, S_{\tau-1}, A_{\tau-1}$ are obtained by executing policy π in the MDP M and τ is the first time when policy π reaches the end state \perp , that is $S_\tau = \perp$ where it always holds that $\tau \leq H$. We use Q^* and V^* to denote the optimal value functions

$$V^*(s) = \sup_{\pi} V^\pi(s), \quad Q^*(s, a) = \sup_{\pi} Q^\pi(s, a), \quad s \in \mathcal{S}, a \in \mathcal{A}$$

We say that the optimal value functions V^* and Q^* can be written as a linear function of d -dimensional features $\psi : \mathcal{S} \sqcup (\mathcal{S} \times \mathcal{A}) \rightarrow \mathbb{R}^d$ if for all state s and action a , $V^*(s) = \langle \theta, \psi(s) \rangle$ and $Q^*(s, a) = \langle \theta, \psi(s, a) \rangle$ for some fixed $\theta \in \mathbb{R}^d$ independent of s and a .² In our construction, linear V^* implies linear Q^* for $\psi(s, a) = \psi(P(s, a))$ as (i) in deterministic transition MDPs, $Q^*(s, a) = r(s, a) + V^*(P(s, a))$, (ii) in our construction, rewards are 0 everywhere except at the leaves and (iii) the reward at the leaves does not depend on the action.

Computational Problems. We next introduce 3-SAT, a satisfiability problem for 3-CNF formulas. In a 3-SAT problem, we are given as input, a 3-CNF formula φ with v variables and $O(v)$ clauses and our goal is to decide if φ is satisfiable.

1. $\Delta([0, 1])$ denotes the set of all distributions over the interval $[0, 1]$.

2. Above, \sqcup means taking the disjoint union of the arguments.

Complexity problem 3-SAT*Input:* A 3-CNF formula φ with v variables and $O(v)$ clauses*Goal:* Decide whether the formula is satisfiable.

The focus of this work is the computational RL problem, LINEAR-k-RL. In a LINEAR-k-RL problem with feature dimension d , we are given access to a deterministic MDP M with k actions and horizon $H = O(d)$ such that the optimal value functions Q^* and V^* can be written as a linear function of the d -dimensional features ψ . Our goal is to output a good policy, which we define as any policy π that satisfies $V^\pi > V^* - 1/8$, where V^π and V^* refers to the value of the policy π and optimal policy, respectively, at a fixed starting state and is always in $[0, H]$ ³. From now on, we always assume that the number of actions is $k = 3$.

Complexity problem LINEAR-k-RL*Oracle:* a deterministic MDP M with k actions, optimal value functions V^* and Q^* linear in d dimensional features ψ , horizon H and state space of size at most $\exp(\text{poly}(d))$.*Goal:* find policy π such that $V^\pi > V^* - 1/8$.

We now describe how the algorithm interacts with the MDP. We assume that the algorithm has access to the state and action spaces (which can be taken as subsets of integers), as well as random access to the associated (i) reward function R , (ii) transition function P and (iii) features ψ . For all these functions, the algorithm provides a state s and action a (if needed) and receives a random sample from the distribution $R(s, a)$ (for the reward function), the state $P(s, a)$ (for the transition function), features $\psi(s)$ and $\psi(s, a)$ (for the features). We assume that each call accrues constant runtime and input/output for these functions are of size polynomial in feature dimension d .

We will often talk about randomized algorithm A solving a problem in time t with error probability p . By this we mean (i) A runs in time $O(t)$; (ii) for satisfiability problems, it returns YES on positive input instances with probability at least $1 - p$ and returns NO on negative input instances with probability 1; and (iii) for an RL problem, it returns a good policy with probability at least $1 - p$.

2.2. Exponential lower bound for LINEAR-3-RL

In this paper, we present computational lower bound under a strengthening of the $\text{NP} \neq \text{RP}$ conjecture, the Randomized Exponential Time Hypothesis (rETH) (Dell et al., 2014), which asserts that probabilistic algorithms can not decide if a given 3-SAT problem with v variables and $O(v)$ clauses is satisfiable in sub-exponential time.

Definition 1 (Randomized Exponential Time Hypothesis (rETH)) *There is a constant $c > 0$ such that no randomized algorithm can decide 3-SAT with v variables in time 2^{cv} with error probability $1/3$.*

3. In our constructions, we satisfy the more stringent condition that $V^* \in [0, 1]$.

The Randomized Exponential Time Hypothesis along with many variants motivated by the Exponential Time Hypothesis (Impagliazzo and Paturi, 2001) has been influential in discovering hardness results for a variety of problems see, e.g. Cygan et al. (2015); Williams (2019). Under the Randomized Exponential Time Hypothesis, our main result is an exponential computational lower bound for learning good policies in deterministic MDPs with linear optimal value functions.

Theorem 1 (Exponential in horizon and dimension lower bound) *Under rETH, there is no randomized algorithm that solves LINEAR-3-RL with feature dimension d and horizon H in time $\exp(\tilde{O}(\min(d^{1/4}, H^{1/4})))$ with probability at least $9/10$, where \tilde{O} hides $\text{polylog}(d)$ and $\text{polylog}(H)$ factors.*

A few remarks are in order. Firstly, $\min(\cdot)$ is the correct complexity measure here. To see this, we note that this problem can be solved in time $\exp(\tilde{O}(\min(d, \sqrt{H})))$ (we prove these upper bounds in Appendix B) and therefore if either dimension d or horizon H is constant, we can solve this problem efficiently in the other parameter. Secondly, this is the first exponential computational lower bound for this setting as the previous best known result Kane et al. (2022) produces at best a quasi-polynomial lower bound, even assuming rETH.

In terms of horizon H , there is still a gap between the $\exp(\tilde{\Omega}(H^{1/4}))$ lower bound in Theorem 1 and the $\exp(\tilde{O}(\sqrt{H}))$ upper bound. We next show a lower bound optimized for horizon H which almost matches this upper bound.

Theorem 2 (Almost matching horizon lower bound) *Under rETH, there is no randomized algorithm that solves LINEAR-3-RL with horizon H and feature dimension $d \geq H^{\log H}$ in time $\exp(\tilde{O}(\sqrt{H}))$ with probability at least $9/10$, where \tilde{O} hides $\text{polylog}(H)$ factors.*

We now discuss some open questions. Even though the lower bound in Theorem 2 almost matches the upper bound in terms of horizon H , it requires the feature dimension to be at least quasi-polynomial in H . We leave it as an open question if the above result also holds when $d = \text{poly}(H)$. Another important direction is understanding the complexity in terms of dimension d i.e. a lower bound optimized for dimension d . Our proof for Theorem 1 can be modified to show $\exp(d)$ lower bound for $H = \exp(d)$. Does the result also hold true for $H = \text{poly}(d)$?

Related Work. We already discussed the large body of work giving statistical efficient algorithms for RL under various assumptions. Complementing them is work giving statistical lower bounds for RL with linear function approximation when the number of actions grows. Concretely, the works of Weisz et al. (2021a, 2022b); Wang et al. (2021) showed sample complexity lower bounds exponential in $\min(d, H)$ that imply statistical hardness of finding a near-optimal policy, when the number of actions grow with the number of basis functions and the values of basis functions are only revealed for the sampled states. Furthermore, there are recent works (Golowich et al., 2022a,b; Uehara et al., 2022) on designing quasipolynomial-time end-to-end algorithm for learning in "observable" POMDPs (our lower bound result refute existence of similar quasipolynomial-time algorithms for linear Q^* and V^* assumption.)

Remainder of this paper. In Section 3, we present a brief overview of the main technical ideas in the lower bound construction. In Section 4, we describe in detail our exponential lower bound constructions and prove our main theorems. In Appendix A we use standard reductions to show that under the randomized Exponential Time Hypothesis, a gap version of SAT that we use in the

reduction is computationally hard. In Appendix B we give algorithms for RL which are exponential in $\min(d, \sqrt{H})$, showing that our lower bound is close to optimal.

3. Proof Overview

The high-level idea of the previous lower bound of Kane et al. (2022) was the following. The authors design an MDP that forces the learner to search for an unknown vector w^* in $\{0, 1\}^v$ which constitutes a satisfying solution of a given SAT formula ϕ . In particular, each state in the MDP corresponds to an assignment and the learner at the state can flip one variable appearing in the first unsatisfying clause of the formula (assuming some canonical ordering of the clauses). Rewards are given when the learner either reaches a satisfying assignment or the end of the horizon. The rewards are designed in such a way that (i) the learner is incentivized for finding w^* quickly but (ii) unable to exploit much information from the rewards to accelerate the searching process. As a result, the task becomes as hard as solving the original SAT problem.

One bottleneck of the above approach is that the reward is only uninformative if the algorithm plays the game for fewer than *quasi-polynomially* many times. After that, there is a decent chance that the algorithm could obtain extra information from the reward structure which may significantly simplify the task. We follow the same high level idea of embedding hard (variants of) SAT instances into a linear-RL problem. Yet, we make significant modifications to the transition and reward structure of the MDP such that the algorithm can hardly obtain any useful information from the rewards unless it plays the game for *exponentially* many times.

In essence, we ensure the rewards given at the end of the horizon are uninformative by making it a Bernoulli variable with exponentially small mean. If so, the learner with high probability sees only 0 in the end unless it plays the game for a large number of times. As a warm-up, one could imagine an MDP with actions and transitions identical to that from Kane et al. (2022). Yet, we modify the reward to be $\exp(-\text{number of steps thus far} - \text{dist}(w, w^*))$ at any terminal state w . This makes sure the (expected) reward given at the end of the horizon is always exponentially small. Unfortunately, the value function induced will be of the same exponential function, and hence cannot be written as a linear function of some low-dimensional features depending only on the state.

Round Based Game. One way to fix this is by turning the game into a round based game. We divide the search into rounds and in each round, the variables are shown sequentially for the learner to decide whether to flip a variable it or not. Then, if the learner terminates at the n -th round, we make the reward function roughly $\prod_{i=1}^n g_i(\text{number of flips taken in round } i) \cdot g_{n+1}(\text{dist}(w, w^*))$ for some carefully chosen low-degree and monotonically decreasing polynomials g_i . Now, consider the greedy policy which tries to decrease the distance to w^* whenever possible. Since the greedy policy can always reach w^* within one (entire) round, the value function of such strategy at the beginning of round i will be $\prod_{j < i} g_j(\text{number of flips in round } j) * g_i(\text{dist}(w, w^*))$. Since only the last term depends on w and w^* , we get that the value function is essentially a low-degree polynomial in w and w^* , which can indeed be written as a linear function of some state-dependent low-dimensional feature vectors. See Lemma 6 for details.

However, in order to ensure that this is the optimal strategy, we will need to define the g_i very precisely so that making a flip in the current round is always better than deferring it to future rounds. Essentially this means that the logarithmic derivative of g_j should be smaller than the logarithmic derivative of g_i for $j > i$. Ideally, we would like to make $g_i(x) = \exp(-c_i x)$ for some increasing sequence of c_i , which would then make the above property trivially true. However, since g_i must be a

Complexity problem (b, ϵ) -GAP-3-SAT

Input: A 3-CNF formula φ with v variables and $O(v)$ clauses with the following promise: (1) each variable is in at most b clauses, and (2) either φ is satisfiable or any assignment leaves at least ϵ -fraction of clauses unsatisfied.

Goal: Decide whether the formula is satisfiable.

polynomial, we will instead make it a Taylor approximation to this exponential function around $x = 0$. As long as we can make the error in this Taylor approximation small relative to the difference in logarithmic derivatives of $\exp(-c_i \cdot x)$, it remains advantageous for the agent to take additional steps in earlier rounds. Fortunately, this is indeed achievable using a low-degree Taylor approximation. See Claim 17 and Lemma 18 for details of the argument.

Flips Enforcement. While the round-based game does ensure the linearity of the value function, the reward given at the end of the horizon is not necessarily small. Since $g_i(x)$ is taken to be the Taylor approximation of $\exp(-c_i \cdot x)$ around $x = 0$, $g_i(0)$ will be 1. Consequently, if the learner chooses to flip nothing, it may receive a huge reward in the end, allowing the algorithm to extract information from the reward structure.

To prevent this, we will offer the learner a bundle of variables in the first step of each round so that it must flip one of the given variables. A caveat of doing so is that we want at least one variable to be indeed erroneous so that flipping it results in the correct truth assignment to it and hence the greedy policy is still well-defined and optimal. Fortunately, this is guaranteed if we simply give the variables appearing in any of the unsatisfied clauses.

This allows us to force the algorithm to make at least one flip. In order to make the rewards diminish at a faster rate, we take the idea further: we keep presenting the learner with unsatisfied clauses involving variables that have not yet been flipped. Only after running out of such clauses, we start to go through the rest of the variables and give the learner the choice to skip flips.

We would like to require that we can find many such clauses. Of course this is not possible to guarantee in a general SAT instance. However, we show there is a special family of 3-SAT instances so that finding assignments where one would quickly run out of such unsatisfied clauses is computationally hard. In particular, we use some standard reductions to show that 3-SAT is approximately as hard as what we call GAP-3-SAT where (a) each variable shows up in a bounded number of clauses (b) if there are no solutions then there are no solutions that satisfy a $(1-\epsilon)$ -fraction of clauses. Note that (b) above implies that it is hard to find any assignment satisfying a $(1-\epsilon)$ -fraction of clauses, and (a) says that flipping a variable can only remove a constant number of unsatisfied clauses from consideration. In particular, if each variable appears in at most b clauses, then any computationally efficient algorithm will never run out of unsatisfied clauses in the first $\epsilon \cdot (\text{total number of clauses}/b)$ steps. This ensures that the reward at the horizon is exponentially small in the number of rounds.

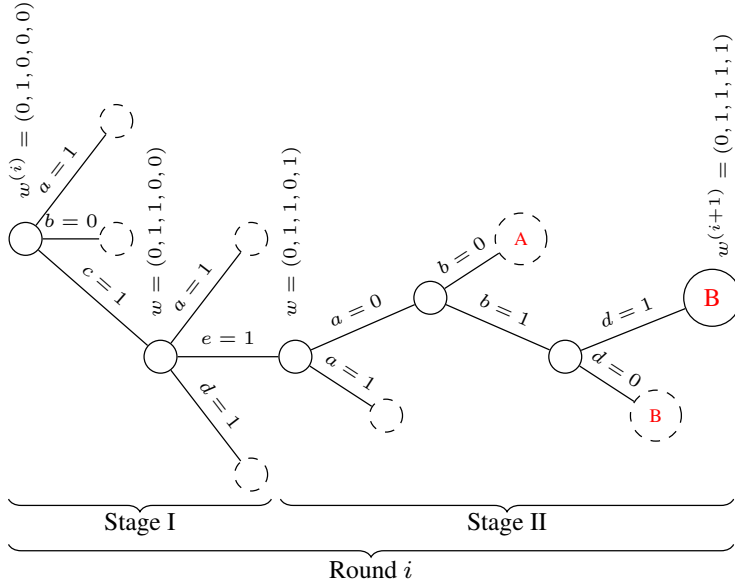


Figure 1: Example mechanics of the MDP for round i . The MDP consists of h consecutive rounds, of which only round i is shown. Nodes are states with their assignment w labeled where it changes, and edges are actions where the label represents the setting of some variable. The satisfiability problem is $(a \vee \neg b \vee c) \wedge (c \vee d \vee e) \wedge (a \vee d \vee e) \wedge (a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg e)$, for variables a to e that have assignment of $w^{(i)}$ at the start of the round. For illustrative simplicity, note that this problem does not belong to (b, ϵ) -GAP-3-SAT. The first two steps form Stage I as there is an unsatisfied clause consisting of only free variables. The second stage allows to change any of the remaining free variables one by one. Transitions are deterministic. Rewards are always zero except for termination conditions **A** and **B**, where the reward is Bernoulli. **A**: the assignment satisfies at least $(1 - \epsilon)$ fraction of clauses. **B**: only if i is the last round, the game is terminated at the end of the round.

4. Lower Bound Construction

In this section, we will prove the following computational lower bound for LINEAR-3-RL under rETH.

Proposition 2 *Let $v \in \mathbb{Z}^+$ be sufficiently large. Suppose $d, H \in \mathbb{Z}^+$ satisfy either (i) $d = v^4 \cdot \text{polylog}(v)$ and $H = \Theta(v^4)$, or (ii) $d = \exp(\log^2 v \cdot \text{polyloglog}(v))$ and $H = \Theta(v^2)$. Then, under rETH, no randomized algorithm can solve LINEAR-3-RL with feature dimension d and time horizon H in time $\exp(v / \text{polylog}(v))$ with error probability $1/10$.*

Our main theorems, Theorem 1 and Theorem 2, follow from Proposition 2 by writing v as a function of d and H .

4.1. From 3-CNF formulas to 3-action MDPs

Recall that in (b, ϵ) -GAP-3-SAT, we are given as input a 3-CNF formula φ on v variables where (1) each variable is guaranteed to occur in at most b clauses and (2) the formula is either satisfiable

or any assignment satisfies at most $(1 - \epsilon)$ -fraction of the clauses (the formula is guaranteed to fall in one of these two cases). Furthermore, we may assume that the number of clauses is at least v . By Proposition 10, we know deciding whether φ is satisfiable must take time that is exponential in v under rETH when b, ϵ are set to be two absolute constants. Our goal is to construct an MDP parametrized by φ so that learning a near optimal policy for the MDP is as hard as solving (b, ϵ) -GAP-3-SAT. When the formula is satisfiable, additionally the MDP will have an extra parameter w^* which is chosen to be an arbitrary satisfying assignment of the formula.

To consolidate the two results in Proposition 2, in our reduction, we construct the MDP with two additional “degree parameters” $p, q \in \mathbb{Z}^+$. The MDP will have feature dimension $d = 2 \cdot v^{2p}$ and time horizon $H = \alpha \cdot v^q$ where α is a small enough absolute constant to be determined later. In particular, the hard instance for the first result is obtained by setting $p = 2, q = 4$ and the hard instance for the second result is obtained by setting $p = 2 \log v, q = 2$.

State Action Transition. The time steps are divided into $h := H/v$ rounds where each round consists of v steps. In each round, the MDP maintains a set of “used variables”, initialized to be the empty set at the beginning of each round. We will call unused variables as “free variables”. One round is further divided into two stages as follows:

1. In each step of the first stage, the agent is given one unsatisfied 3-SAT clause with only *free variables* and asked which of the three variables should be flipped. Then, the variable chosen by the algorithm will be added to the set of used variables. The first stage ends (and the second stage starts) when we run out of unsatisfied clauses with only free variables.
2. In the second stage, the MDP presents each of the remaining free variables sequentially to the agent and asks whether the variable should be flipped. After each step, regardless of whether the algorithm decides to flip the variable or not, the variable presented will be added to the set of used variables.

Note that each round has exactly v steps since there are v variables in the formula and each step marks one of them as used.

Termination Condition. The MDP terminates if it reaches the last level, or if more than a $(1 - \epsilon)$ -fraction of the clauses are satisfied. We make a couple of observations related to the termination conditions. First, if the MDP terminates before reaching the last level, the algorithm has essentially solved the underlying GAP-3-SAT problem since this means there exists an assignment that satisfies more than $(1 - \epsilon)$ -fraction of the clauses. Secondly, the termination condition ensures that, at the beginning of each round, there are at least an ϵ -fraction of unsatisfied clauses. Since each variable appears in at most b clauses, we will never run out of unsatisfied clauses with only free variables in the first $\epsilon \cdot (\text{total number of clauses}/b)$ steps.

Size of state space. The number of states necessary for a round and an assignment is at most $O(3^v)$: the transitions within a round form a tree of branching factor at most 3 and height at most v (see fig. 1). There are $h = H/v$ rounds, the whole transition structure is a tree, hence the total number of states is at most $O((3^v)^h) = \exp(\text{poly}(v, H))$.

Rewards. Rewards are given only when the MDP terminates and are different depending on whether the formula is satisfiable or not. When the formula is not satisfiable, the reward is 0 everywhere. In the rest of the discussion, we will think of assignments as vectors in $\{-1, 1\}^v$.

When the formula is satisfiable, we need to keep track of the assignment at the beginning of each round and denote them as $w^{(1)}, w^{(2)} \dots, w^{(n)}$ (note that the algorithm starts at the first state with the assignment $w^{(1)}$), on which the final reward depends. The reward depends only on the history $w^{(1)}, w^{(2)} \dots, w^{(n)}$, the current assignment w and the optimal assignment w^* and is given by $Ber(r(w^{(1)}, w^{(2)} \dots, w^{(n)}, w))$ where $r(\cdot)$ is the expected reward function and the Bernoulli distribution $Ber(\rho)$ is 1 with probability ρ and 0 with probability $1 - \rho$. Before specifying the expected reward function $r(\cdot)$, we introduce the concept of an *extended assignment*.

Definition 3 (Extended Assignment) *Let S be the set of free variables. Then, the extended assignment of w under S , denoted as $\text{ext}(w, S)$ is given by $\text{ext}(w, S)_i = w_i^*$ for $i \in S$ and $\text{ext}(w, S)_i = w_i$ for $i \notin S$.*

In plain language, the extended assignment is the assignment derived from w after correcting all the free variables to agree with w^* . We note that dependence of the reward function on the extended assignment is crucial to ensure that the value functions associated to the greedy policy are linear functions, which will become relevant later. Now we are ready to define expected reward function $r(\cdot)$ in terms of the historic assignments $w^{(1)}, \dots, w^{(n)}$ reached by the agent at the end of past rounds, the current assignment w reached by the agent and the set of free variables S when the MDP terminates.

Definition 4 (Expected Reward) *Let $p, q \in \mathbb{Z}^+$ be the two degree parameters. Let $T_p : \mathbb{R} \mapsto \mathbb{R}^+$ be the degree- p Taylor approximation of the exponential function $\exp(\cdot)$ at zero:*

$$T_p(x) = \sum_{i=0}^p \frac{x^i}{i!}.$$

Then, we define expected reward function $r(w^{(1)}, w^{(2)} \dots, w^{(n)}, w, S)$ as

$$\left(\prod_{i=1}^{n-1} g_i(\text{dist}(w^{(i)}, w^{(i+1)})) \right) \cdot g_n(\text{dist}(w^{(n)}, \text{ext}(w, S))) \cdot g_{n+1}(\text{dist}(\text{ext}(w, S), w^*)), \quad (1)$$

where the polynomial $g_i : \mathbb{R} \mapsto \mathbb{R}$ for round i is defined as

$$g_i(x) = T_p \left(-\frac{x}{v^{q-1} \cdot (3 - i/h)} \right). \quad (2)$$

As noted in the proof overview, the polynomials g_i are chosen to ensure that the optimal policy prefers going towards w^* as fast as possible and using a low degree Taylor approximation ensures the value function for the optimal policy can be written as a linear function of low dimensional features.

4.2. Linear Value Function

When the underlying formula is unsatisfiable, any policy is optimal since the reward is constantly 0. When the formula is satisfiable, we will show that the “greedy policy” is optimal.

Definition 5 (Greedy Policy) *We say a policy is greedy if at every state it chooses any action that decreases the distance to w^* whenever possible. If not, it tries to not increase the distance to w^* .*

Notice that based on our setup of the MDP greedy policies exist: in the first stage of a round, the algorithm is given an unsatisfied clause so there is at least one variable in the clause that can be flipped to decrease the distance from the current assignment to w^* ; in the second stage, the algorithm is given variables one at a time and it can always choose to not flip the variable if the current assignment already agrees with w^* on the variable.

We first discuss the value function V^π associated to a greedy policy π . Given a state with current assignment w and a set S of free variables, we define the following concepts that will be useful in the discussion. Let $m(S) \in \{0, 1\}^v$ be the masking vector such that $m(S)_i = 1$ if the i -th variable is in S and $m(S)_i = 0$ otherwise. Moreover, let $\mathbf{1}$ denote the all-one vector and \circ the point-wise multiplication operator. Then, we define

$$\begin{aligned} \text{dist}_{S,\text{free}}(w, w^*) &= \text{dist}(w \circ m(S), w^* \circ m(S)) \\ \text{dist}_{S,\text{used}}(w, w^*) &= \text{dist}(w \circ (\mathbf{1} - m(S)), w^* \circ (\mathbf{1} - m(S))) \end{aligned}$$

In other words, $\text{dist}_{S,\text{used}}(w, w^*)$ and $\text{dist}_{S,\text{free}}(w, w^*)$ are the number of used and free variables respectively where the current assignment differs from w^* . Note that $\text{dist}_{S,\text{used}}(w, w^*) + \text{dist}_{S,\text{free}}(w, w^*) = \text{dist}(w, w^*)$.

Moreover, since the Hamming distance $\text{dist}(a, b)$ for two vectors $a, b \in \{-1, 1\}^v$ is linear in both a and b (as $\text{dist}(a, b) = (v - \langle a, b \rangle)/2$), this implies $\text{dist}_{S,\text{free}}(w, w^*)$ and $\text{dist}_{S,\text{used}}(w, w^*)$ can be written as a linear function of w^* and some state specific parameters depending on the current assignment w and the set of free variables S only. This allows us to show that the value functions for the greedy policy can also be written as linear functions of w^* and some state specific parameters.

Lemma 6 *When φ is satisfiable, the greedy policy’s value at state s with round history $w^{(1)}, \dots, w^{(n)}$, current assignment w and the set of free variables S , is given by*

$$V^\pi(s) = \prod_{i=1}^{n-1} g_i(\text{dist}(w^{(i)}, w^{(i+1)})) \cdot g_n(\text{dist}(w^{(n)}, w) + \text{dist}_{S,\text{free}}(w, w^*)) \cdot g_{n+1}(\text{dist}_{S,\text{used}}(w, w^*)). \quad (3)$$

As a result, there exists features $\psi(s), \psi(s, a) \in \mathbb{R}^d$ with feature dimension $d \leq 2v^{2p}$ depending only on state s and action a ; and $\theta \in \mathbb{R}^d$ depending only on w^ such that V^π and Q^π can be written as a linear function of features ψ i.e. $V^\pi(s) = \langle \theta, \psi(s) \rangle$ and $Q^\pi(s, a) = \langle \theta, \psi(s, a) \rangle$.*

We present the proof in Appendix C. Using this we show the main structural result of this section: the optimality of the greedy policy.

Lemma 7 *Any greedy policy as defined in Definition 5 is optimal.*

We present the proof in Appendix C.1.

4.3. RL algorithm to SAT algorithm

Following the approach taken in previous lower bound Kane et al. (2022), we now build a randomized algorithm \mathcal{A}_{SAT} for 3-SAT using a randomized algorithm \mathcal{A}_{RL} for the RL problem. In particular, we build an “approximate” simulator \bar{M}_φ for the MDP oracle M_φ . The simulator \bar{M}_φ is exactly the MDP M_φ in terms of the transition function and features associated with the MDP M_φ , but differs in the reward function at the last layer which is always 0 for the simulator \bar{M}_φ . With the proposed modification, we can execute each call to simulator \bar{M}_φ in time $\text{poly}(d)$.

Algorithm. On input 3-CNF formula φ , \mathcal{A}_{SAT} runs the algorithm \mathcal{A}_{RL} replacing each call to MDP oracle M_φ with the corresponding call to simulator \bar{M}_φ . Recall that the output for the RL algorithm in our setting (deterministic transition MDP) is a sequence of actions. If the sequence of actions returned by \mathcal{A}_{RL} ends on a state with an assignment w that satisfies more than $(1 - \epsilon)$ -fraction of the clauses, \mathcal{A}_{SAT} terminates the simulation immediately and outputs YES. If \mathcal{A}_{RL} throughout the simulation never finds any state associated with such an assignment, \mathcal{A}_{SAT} outputs NO.

Correctness. To complete our reduction, we will show the following:

- (i) If algorithm \mathcal{A}_{RL} outputs a policy π such that $V^\pi > V^* - 1/8$, then \mathcal{A}_{SAT} on 3-CNF formula φ outputs YES if φ is satisfiable and NO otherwise.
- (ii) If \mathcal{A}_{RL} with access to MDP oracle M_φ outputs a policy π such that $V^\pi > V^* - 1/8$ with error probability $1/10$, then \mathcal{A}_{RL} with access to simulator \bar{M}_φ outputs a policy π such that $V^\pi > V^* - 1/8$ with respect to M_φ with error probability $1/8$ (namely, even though \mathcal{A}_{RL} is interacting with the simulator \bar{M}_φ , the returned policy is guaranteed to do well on the true MDP M_φ).

Recalling that if ϕ is not satisfiable, any policy is optimal, the above two claims establish that \mathcal{A}_{SAT} solves GAP-3-SAT with error probability $\leq 1/8$. We start by proving that if \mathcal{A}_{RL} succeeds on MDP M_φ , then \mathcal{A}_{SAT} succeeds on 3-CNF formula φ . This follows from the fact that any good policy in the MDP M_φ must reach a state with the assignment w^* , the satisfying assignment which is arbitrarily chosen to construct M_φ .

Proposition 8 *Assume that α, b, ϵ are constants and that v is large enough. Then, if φ is satisfiable and \mathcal{A}_{RL} running on M_φ returns a policy π satisfying $V^\pi > V^* - 1/8$ then π ends on an assignment that satisfies at least a $(1 - \epsilon)$ -fraction of clauses.*

The detailed proof can be found in Appendix C.2. Next, we show that the behavior of \mathcal{A}_{RL} is about the same even if it is run on the simulator \bar{M}_φ . In particular, given \mathcal{A}_{RL} runs in sub-exponential time and succeeds on M_φ , we could argue \mathcal{A}_{RL} will be provided about the same information when it is executed on \bar{M}_φ and on M_φ and therefore would succeed on the outputs of simulator \bar{M}_φ albeit with a smaller constant probability.

Proposition 9 *Suppose \mathcal{A}_{RL} with access to MDP oracle M_φ runs in time T and outputs a policy π such that $V^\pi > V^* - 1/8$ with error probability $1/10$. Further, assume that the expected reward at the last layer of M_φ is upper bounded by $1/(5T)$. Then \mathcal{A}_{RL} with access to simulator \bar{M}_φ , still running in time T , outputs a policy π such that $V^\pi > V^* - 1/8$ with respect to M_φ with error probability $1/8$.*

The detailed proof can be found in Appendix C.2. We next prove using standard reductions that (b, ϵ) - GAP-3-SAT is approximately as hard as 3-SAT.

Proposition 10 *Under rETH, there exists constants $b, \epsilon, c > 0$ such that no randomized algorithm can solve (b, ϵ) - GAP-3-SAT with v variables in time $\exp(cv/\text{polylog}(v))$ with error probability $1/8$.*

We provide a proof in Appendix A. Now, we are ready to prove our main result, Proposition 2. For this, we demonstrate how one could reduce a (b, ϵ) -GAP-3-SAT instance into an MDP instance.

Proof [Proof of Proposition 2] Set $p = 2, q = 4$ or $p = 2 \log v$ and $q = 2$. For any $v \in \mathbb{Z}^+$, suppose there exists an algorithm \mathcal{A}_{RL} which can solve LINEAR-3-RL with feature dimension $d = \Theta(v^{2p})$ and $H = \Theta(v^q)$ with error probability $1/10$ and runs in time $\exp(c_1 \cdot v / \text{polylog}(v))$ for $c_1 < \min(1/2, c/2)$ where c is the constant from Proposition 10. Then, we claim we can build another algorithm \mathcal{A}_{SAT} which can solve (b, ϵ) -GAP-3-SAT with error probability $1/8$ in time $\exp(cv / \text{polylog}(v))$. Note that this would contradict Proposition 10 under rETH and hence prove our proposition.

Let φ be the 3-CNF formula of a (b, ϵ) -GAP-3-SAT instance containing v variables and at least v clauses. Then, by definition, each variable appears in at most b clauses. Furthermore, φ is guaranteed to either be satisfiable or that at least an ϵ -fraction of the clauses are not satisfiable under any assignment. To decide between the two cases, we first build an MDP M_φ (parameterized by the two positive integers p, q) as described in Section 4.1. In particular, the MDP is designed to have αv^{q-1} rounds and the polynomials g_i will be a degree- p Taylor approximations as specified in Equation (2). As α, b, ϵ are absolute constants, we ignore the dependence on them below.

We will proceed to bound the time horizon and the feature dimension of M_φ respectively. Since each round consists of v steps, the horizon is $H = \Theta(v^q)$. Furthermore, by Lemma 6, the value function for the greedy policy can be written as a linear function of a feature vector of size $\Theta(v^{2p})$. By Lemma 18, the greedy policy is optimal. Hence, the feature dimension of the MDP is $d = \Theta(v^{2p})$.

Next, as noted in the proof of Proposition 8, for any policy π which terminates on the last level, the expected reward is always upper bounded by $\exp(-v)$. Let \bar{M}_φ be the MDP that differs from M_φ only with respect to the rewards received at the end of the horizon (the rewards of \bar{M}_φ are consistently 0). Then, by Proposition 9 and small $\exp(-v)$ reward noted above, we know \mathcal{A}_{RL} , when ran for at most $\exp(c_1 \cdot v / \text{polylog}(v))$ time (as $c_1 < 1/2$) on the simulator of \bar{M}_φ , will still output a good policy $\tilde{\pi}$ with respect to M_φ with probability at least $7/8$.

By Proposition 8, if \mathcal{A}_{RL} succeeds and φ is satisfiable, then the policy $\tilde{\pi}$ will terminate on a satisfying assignment. Hence, we can just check the path obtained by running policy $\tilde{\pi}$ to decide whether φ is satisfiable, which takes at most $\text{poly}(v)$ time.

Hence, the existence of such an algorithm \mathcal{A}_{RL} which runs in time at most $\exp(c_1 \cdot v / \text{polylog}(v))$ time implies the existence of another algorithm which can solve the (b, ϵ) -GAP-3-SAT problem in time $\exp(c_1 v / \text{polylog}(v)) + \text{poly}(v) \leq \exp(cv / \text{polylog}(v))$. ■

Acknowledgments

Daniel Kane acknowledges support under NSF Award CCF-1553288 (CAREER) and a Sloan Research Fellowship. Shachar Lovett acknowledges support under NSF Awards DMS-1953928 and CCF-2006443. Csaba Szepesvári gratefully acknowledges the funding from NSERC, Amii, and the Canada AI Research Chair program.

References

Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. *arXiv preprint arXiv:2006.10814*, 2020.

- Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin F Yang. Model-based reinforcement learning with value-targeted regression. *arXiv:2006.01107*, 2020.
- Marek Cygan, Fedor Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Lower Bounds Based on the Exponential-Time Hypothesis*, pages 467–521. 07 2015. ISBN 978-3-319-21274-6. doi: 10.1007/978-3-319-21275-3_14.
- Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslamán, and Martin Wahlén. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4), 2014.
- Shi Dong, Benjamin Van Roy, and Zhengyuan Zhou. Provably efficient reinforcement learning with aggregated states, 2020.
- Simon Du, Sham Kakade, Jason Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in rl. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 2826–2836, 18–24 Jul 2021.
- Simon S Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudík, and John Langford. Provably efficient RL with rich observations via latent state decoding. In *International Conference on Machine Learning*, 2019.
- Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020a.
- Simon S Du, Jason D Lee, Gaurav Mahajan, and Ruosong Wang. Agnostic Q-learning with function approximation in deterministic systems: Tight bounds on approximation error and sample complexity. In *Advances in Neural Information Processing Systems*, 2020b.
- Dylan J Foster, Sham M Kakade, Jian Qian, and Alexander Rakhlin. The statistical complexity of interactive decision making. *arXiv preprint arXiv:2112.13487*, 2021.
- Noah Golowich, Ankur Moitra, and Dhruv Rohatgi. Learning in observable pomdps, without computationally intractable oracles. *arXiv preprint arXiv:2206.03446*, 2022a.
- Noah Golowich, Ankur Moitra, and Dhruv Rohatgi. Planning in observable pomdps in quasipolynomial time. *arXiv preprint arXiv:2201.04735*, 2022b.
- Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.
- Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low bellman rank are pac-learnable, 2016.
- Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, 2020.
- Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. *arXiv preprint arXiv:2102.00815*, 2021.

- Daniel Kane, Sihan Liu, Shachar Lovett, and Gaurav Mahajan. Computational-statistical gaps in reinforcement learning. *arXiv preprint arXiv:2202.05444*, 2022.
- Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Pac reinforcement learning with rich observations. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1848–1856, 2016.
- Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, 2020.
- Lihong Li. *A Unifying Framework for Computational Reinforcement Learning Theory*. PhD thesis, USA, 2009. AAI3386797.
- Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. In *NIPS*, volume 14, page 30, 2001.
- Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. Sample complexity of reinforcement learning using linearly combined model ensembles. In *Conference on Artificial Intelligence and Statistics*, 2020.
- Dana Moshkovitz and Ran Raz. Two-query pcp with subconstant error. *Journal of the ACM (JACM)*, 57(5):1–29, 2008.
- Rémi Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1006. Menlo Park, CA; Cambridge, MA; London; AAI Press; MIT Press; 1999, 2005.
- Christos H Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of computer and system sciences*, 43(3):425–440, 1991.
- Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, 2019.
- Masatoshi Uehara, Ayush Sekhari, Jason D Lee, Nathan Kallus, and Wen Sun. Computationally efficient pac rl in pomdps with latent determinism and conditional embeddings. *arXiv preprint arXiv:2206.12081*, 2022.
- Yuanhao Wang, Ruosong Wang, and Sham M. Kakade. An exponential lower bound for linearly-realizable mdps with constant suboptimality gap, 2021.
- Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory*, pages 1237–1264. PMLR, 2021a.
- Gellért Weisz, András György, Tadashi Kozuno, and Csaba Szepesvári. Confident approximate policy iteration for efficient local planning in q^π -realizable mdps. In *Advances in Neural Information Processing Systems*, 2022a.

- Gellért Weisz, Csaba Szepesvári, and András György. Tensorplan and the few actions lower bound for planning in mdps under linear realizability of optimal value functions. In *International Conference on Algorithmic Learning Theory*, pages 1097–1137. PMLR, 2022b.
- Gellért Weisz, Philip Amortila, Barnabás Janzer, Yasin Abbasi-Yadkori, Nan Jiang, and Csaba Szepesvári. On query-efficient planning in mdps under linear realizability of the optimal state-value function, 2021b.
- Zheng Wen and Benjamin Van Roy. Efficient reinforcement learning in deterministic systems with value function generalization. *Mathematics of Operations Research*, 42(3):762–782, 2017.
- Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. *Proceedings of the International Congress of Mathematicians (ICM 2018)*, 2019.
- Lin Yang and Mengdi Wang. Sample-optimal parametric Q-learning using linearly additive features. In *International Conference on Machine Learning*, 2019.
- Dong Yin, Botao Hao, Yasin Abbasi-Yadkori, Nevena Lazić, and Csaba Szepesvári. Efficient local planning with linear function approximation. In *International Conference on Algorithmic Learning Theory*, pages 1165–1192. PMLR, 2022.
- Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent bellman error, 2020.
- Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted mdps with feature mapping. In *International Conference on Machine Learning*, pages 12793–12802, 2021.

Contents

1 Introduction **2**

2 Our Contributions **3**

 2.1 Preliminaries 3

 2.2 Exponential lower bound for LINEAR-3-RL 4

3 Proof Overview **6**

4 Lower Bound Construction **8**

 4.1 From 3-CNF formulas to 3-action MDPs 8

 4.2 Linear Value Function 10

 4.3 RL algorithm to SAT algorithm 11

A Hardness of Approximate SAT with gap and few clauses **17**

B Upper Bounds **19**

C Omitted Proof from Section 4 **23**

 C.1 Optimality of Greedy Policy 24

 C.2 Reduction from GAP-3-SAT to LINEAR-k-RL 29

Appendix A. Hardness of Approximate SAT with gap and few clauses

In this section, we prove the following:

Proposition 11 *Under rETH, there exists constants $b, \epsilon, c > 0$ such that no randomized algorithm can solve (b, ϵ) - GAP-3-SAT with v variables in time $\exp(cv / \text{polylog}(v))$ with error probability $1/8$.*

To prove this, we will look at another problem: ϵ - GAP-3-SAT. This is similar to (b, ϵ) - GAP-3-SAT except it does not put any constraints on how many clauses a variable can be in. Through standard technique, one can show that ϵ - GAP-3-SAT is also hard. In particular, its hardness is shown in [Moshkovitz and Raz \(2008\)](#) and relies on a certain version of the Probabilistic Checkable Proof (PCP) theorem.

Complexity problem ϵ - GAP-3-SAT

Input: A gap parameter $\epsilon > 0$ and a 3-CNF formula φ with v variables and $O(v)$ clauses such that the either (i) φ is satisfiable or (ii) any assignment leaves at least an ϵ -fraction of the clauses unsatisfied where $\epsilon > 0$.

Goal: Decide whether the formula is satisfiable.

Theorem 3 (Reduction from 3-SAT to GAP-3-SAT) *Solving 3-SAT on inputs of size n can be reduced to distinguishing between the case that a 3-CNF formula of size $n \cdot \text{polylog}(n)$ is satisfiable and the case that only $1 - \epsilon$ fraction of its clauses are satisfiable for some constant $\epsilon > 0$.*

For completeness, we provide a proof for the above theorem. We first review some basic concepts about the PCP theorem. Given a statement (for example, whether a SAT instance is satisfiable), a PCP verifier is granted query access to a proof constructed for the statement over an alphabet Σ and asked to decide whether the statement is true. A PCP verifier has several important parameters.

- **Completeness c :** The minimal probability that the verifier accepts a correct proof.
- **Soundness ϵ :** The maximal probability that the verifier accepts a proof for an incorrect theorem.
- **Queries q :** The number of queries made by the verifier to the proof.
- **Size m :** The length of the proof.
- **Randomness r :** The number of random bits used by the verifier.
- **Alphabet Σ :** The alphabet used by the proof.

We denote by $\text{PCP}_{c,\epsilon}[r, q]_{\Sigma}$ the class of languages that have a PCP verifier with completeness c , soundness s , randomness r , and q queries to a proof over alphabet Σ . Moreover, the PCP verifier is only allowed to do a two query projection test. In a two query-projection test, the verifier is only allowed to make two queries. Upon seeing the answer to the first query, the verifier either immediately rejects, or it has uniquely determined answer to the second query on which it accepts. Our starting point is the following theorem from [Moshkovitz and Raz \(2008\)](#).

Theorem 4 (Theorem 7 from [Moshkovitz and Raz \(2008\)](#)) *There exists a constant $\epsilon > 0$ and an alphabet Σ of constant size, such that $3\text{-SAT} \in \text{PCP}_{1,1-\epsilon}[\log n + O(\log \log n), 2]_{\Sigma}$.*

Proof [Proof of Theorem 3] Given a 3-CNF formula ψ with size n , the goal is to use the verifier in the above theorem to construct a different 3-SAT instance ϕ with size $O(n \text{ polylog } n)$ such that (i) ϕ is satisfiable if ψ is satisfiable; and (ii) at least an ϵ fraction of the clauses in ϕ are not satisfiable under any assignment if ψ is not satisfiable.

Notice that we can without loss of generality assume the verifier is deterministic if we assume it also takes $r := \log n + O(\log \log n)$ random bits as input. Fix a random bit string, the verifier reads at most 2 characters from the proof. Since there are at most $2^r = n \text{ polylog } n$ different random bit strings, we can without loss of generality assume the proof is of size at most $T := 2n \text{ polylog } n$. The first step of the construction is to create T variables $\{x_1, x_2, \dots, x_T\}$ where $x_i \in \Sigma$ represent the queries responses given to the verifier. We will create a SAT formula for each of the 2^r random bit strings and the final construction will be simply the concatenation of all the SAT formulas with the “AND” logical operator. Fix an arbitrary random string $q \in \{0, 1\}^r$. We can then compute the first position the verifier will read. We can denote it as $l_1(q)$. There will be a subset of values $R(q) \subseteq \Sigma$ that the verifier will reject immediately $x_{l_1(q)} \in R(q)$. If the verifier does not reject immediately, the verifier could branch off to do different things based on the value of $x_{l_1(q)}$. Suppose, $x_{l_1(q)} = \sigma \in \Sigma \setminus R(q)$. We can then compute the second position the verifier will read, which we denote as

$l_2(q, \sigma)$, and the “right” character the verifier is expecting, which we denote as $f(q, \sigma)$. Then, we know that the verifier will accept if and only if the proof, represented by x_1, \dots, x_T satisfies that

$$\bigcup_{\sigma \in \Sigma \setminus R} (x_{l_1(q)} = \sigma) \wedge (x_{l_2(q, \sigma)} = f(q, \sigma)).$$

Since Σ is of constant size, it is easy to see that one can use a binary encoding for Σ and convert the above statement into a 3-CNF formula of constant size. In addition, there are at most $n \text{ polylog } n$ binary strings q . Hence, the overall 3-CNF formula ϕ is of size $O(n \text{ polylog } n)$. We know that the verifier would reject with probability at least ϵ if the original sat instance ψ is not satisfiable. Hence, at least an ϵ fraction of the sub-formulas of ϕ will not be satisfied under any assignment (which can be interpreted as the binary encoding of the given proof). On the other hand, if ψ is satisfiable, it then holds every sub-formula of ϕ is satisfiable since the verifier always accepts under the “correct” proof.

If one has an algorithm which can distinguish between the cases that ϵ -fraction of ϕ cannot be satisfied under any assignment versus ϕ is satisfiable, one can then decide the satisfiability of ψ . ■

We are interested in (b, ϵ) -GAP-3-SAT, which is a restricted version of ϵ -GAP-3-SAT, where each variable is promised to appear in at most b clauses for some constant b . One can show that approximating (b, ϵ) -GAP-3-SAT is also hard through a reduction given in [Papadimitriou and Yannakakis \(1991\)](#).

Proposition 12 (Adapted from Proof of Theorem 2 in [Papadimitriou and Yannakakis \(1991\)](#))

For some constant integer $b = O(1)$, there is a polynomial time transformation which maps a 3-CNF formula ϕ to another 3-CNF formula ψ over the same set of variables such that

1. *Each variable appears in at most b clauses in ψ .*
2. *If ϕ is satisfiable, then ψ is also satisfiable.*
3. *Let $|\phi|, |\psi|$ denotes the number of clauses in ϕ, ψ respectively. Then, $|\phi| \leq |\psi| \leq O(1) \cdot |\phi|$.*
4. *Let $\max(\phi), \max(\psi)$ denote the maximum number of clauses satisfiable in ϕ and ψ respectively. It holds $\max(\psi) \leq \max(\phi) + |\psi| - |\phi|$.*

Proof [Proof of Proposition 11] Proposition 12 states there is an efficient algorithm translating an ϵ -GAP-3-SAT instance consisting of m clauses into an $(b, \alpha \cdot \epsilon)$ -GAP-3-SAT instance for some constant $\alpha \in (0, 1)$. Hence, if there is no sub-exponential algorithm for the computational problem ϵ_1 -GAP-3-SAT for some constant $\epsilon_1 \in (0, 1)$, there is no sub-exponential algorithm for (b, ϵ_2) -GAP-3-SAT either for some constant b and $\epsilon_2 \in (0, \epsilon_1)$. Combining Theorem 3 and Proposition 12 proves our claim. ■

Appendix B. Upper Bounds

In both of the upper bounds, the final policy computed by our algorithms is of the following form: at the state s , we have some estimations $\tilde{Q}(s, a)$ for each $a \in \mathcal{A}$ such that $|\tilde{Q}(s, a) - Q^*(s, a)| \leq \epsilon$ and the policy always chooses the action $a = \operatorname{argmax}_a \tilde{Q}(s, a)$. We claim the policy induced is nearly optimal as long as ϵ is sufficiently small. The formal statement is given below.

Lemma 13 For any state action pair (s, a) , let $\tilde{Q}(s, a)$ be an approximation of $Q^*(s, a)$ satisfying $|\tilde{Q}(s, a) - Q^*(s, a)| \leq \epsilon/(2H)$. Then, consider the policy π such that at the state s , it always chooses the action $a = \operatorname{argmax}_a \tilde{Q}(s, a)$. Then, it holds $V^\pi(s) \geq V^*(s) - \epsilon$ for any state.

Proof We claim π is a policy satisfying that $V^\pi(s) \geq V^*(s) - \epsilon \cdot h/H$ for any state s in the MDP such that there are still h steps remaining. We show this via induction on the number of steps remaining. Suppose s is a state right before the last step. Then, $V^*(s) = Q^*(s, a^*)$ for $a^* = \operatorname{argmax}_a Q^*(s, a)$ and $V^\pi(s) = Q^*(s, a')$ for $a' = \operatorname{argmax}_a \tilde{Q}(s, a)$. By our assumption, we have $V^\pi(s) \geq \tilde{Q}(s, a') - \epsilon/(2H)$ and since $a' = \operatorname{argmax}_a \tilde{Q}(s, a)$, we then further have

$$V^\pi(s) \geq \tilde{Q}(s, a^*) - \epsilon/(2H) \geq Q^*(s, a^*) - \epsilon/H = V^*(s) - \epsilon/H.$$

Now, consider a state s such that there are $(h + 1)$ steps remaining. Still, let $a' = \operatorname{argmax}_a \tilde{Q}(s, a)$ and $a^* = \operatorname{argmax}_a Q^*(s, a)$. Furthermore, let $s' = P(s, a')$ be the next state after applying a' . We then have $V^*(s) = Q^*(s, a^*)$ and $V^\pi(s) = \mathbb{E}[R(s, a')] + V^\pi(s')$. We then have

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R(s, a')] + V^\pi(s') && \text{(Definition of the policy } \pi) \\ &\geq \mathbb{E}[R(s, a')] + V^*(s') - \epsilon h/H && \text{(Inductive Hypothesis)} \\ &= Q^*(s, a') - \epsilon h/H && \text{(Definition of } Q^*, V^*) \\ &\geq \tilde{Q}(s, a') - \epsilon h/H - \epsilon/(2H) && \text{(Assumption about } \tilde{Q}) \\ &\geq \tilde{Q}(s, a^*) - \epsilon h/H - \epsilon/(2H) && \text{(The choice of } a' = \operatorname{argmax}_a \tilde{Q}(s, a)) \\ &\geq Q^*(s, a^*) - \epsilon h/H - \epsilon/H && \text{(Assumption about } \tilde{Q}) \\ &= V^*(s) - \epsilon(h + 1)/H && \text{(The choice of } a^* = \operatorname{argmax}_a Q^*(s, a)) \end{aligned}$$

This then gives us $V^\pi(s) \geq V^*(s) - \epsilon$ for any state since there are in total H steps in the MDP. \blacksquare

We first prove a computational upper bound which is exponential in the feature dimension d . On a high level, we discretize the parameter space that θ^* may lie in to create a policy cover which allows us to search for the best in class by estimating the value of each policy.

Proposition 14 Assume the MDP has a constant number of actions, optimal value functions V^* and Q^* linear in d dimensional features ψ , time Horizon H and state space at most $\exp(\text{poly}(d))$. Furthermore, assume the feature vectors satisfy $\|\psi(s, a)\|_2 \leq 1$ for all state action pairs and $\|\theta^*\|_2 \leq 1$ for the optimal policy parameter θ^* . Let $\epsilon \in (0, 1)$. There is an algorithm which takes $\exp(c \cdot d \cdot \log(Hd/\epsilon))$ time for some sufficiently large constant c and finds a policy π such that $V^\pi(s) \geq V^*(s) - \epsilon$ with probability $9/10$.

Proof Let θ^* denote the unknown parameters of the optimal Q^* function, i.e. $Q^*(s, a) = \langle \theta^*, \psi(s, a) \rangle$. Suppose we can find such a $\theta \in \mathbb{R}^d$ satisfying $\|\theta - \theta^*\|_2 \leq \epsilon/(2H \cdot \sqrt{d})$. We note that this implies

$$|\langle \theta, \psi(s, a) \rangle - Q^*(s, a)| \leq \epsilon/(2H). \quad (4)$$

Then, consider the policy $\pi(\theta)$ such that at state s it always chooses the action $a = \operatorname{argmax}_a \langle \theta, \psi(s, a) \rangle$. By Lemma 13, it holds $V^{\pi(\theta)}(s) \geq V^*(s) - \epsilon$ for any state s .

Now, let $S \in \mathbb{R}^d$ be the set of vectors that form an $\epsilon/(2H\sqrt{d})$ -cover of the d -dimensional unit sphere, i.e. $\min_{\theta \in S} \|\theta - \theta^*\|_2 \leq \epsilon/(2H\sqrt{d})$ for any θ^* satisfying $\|\theta^*\|_2 \leq 1$. Through a standard

combinatorial construction, there exists such a cover S with size $|S| \leq \exp(c \cdot d \cdot \log(Hd/\epsilon))$ for some sufficiently large constant c . From the argument above, we know there must be some $\theta \in S$ such that $\pi(\theta)$ is nearly optimal, i.e. $V^{\pi(\theta)}(s) \geq V^*(s) - \epsilon$ for any state s .

Our strategy is simple: we will try $\pi(\theta)$ for all $\theta \in S$ in a brute-force manner and estimate the expected reward of the induced trajectory up to accuracy ϵ . Notice that the maximum reward collected by any trajectory is at most H . Hence, if we visit the same trajectory with $\text{poly}(H, 1/\epsilon) \cdot \log(1/\delta)$ many times, we can then compute an estimation of its expected reward up to accuracy ϵ with probability at least $1 - \delta$. We can take $\delta = \frac{1}{|S|}$ so that by union bound our estimation for $V^{\pi(\theta)}(s_0)$ is accurate up to error ϵ for all $\theta \in S$ with probability at least $9/10$. Condition on that, we can then choose θ such that it maximizes our empirical estimations of $V^{\pi(\theta)}(s_0)$. Then, it is easy to see that such a $\pi(\theta)$ must satisfy $V^{\pi(\theta)}(s_0) \geq V^*(s_0) - 2\epsilon$. Now, since to simulating the interaction of one trajectory takes time at most $\text{poly}(d, H)$, the total runtime is bounded by

$$\text{poly}(d, H) \cdot \text{poly}(H, 1/\epsilon) \cdot \log(1/|S|) \cdot |S| \leq \exp(c \cdot d \cdot \log(Hd/\epsilon))$$

for some sufficiently large constant c . ■

To prove a horizon upper bound, we build on results of previous work (Du et al., 2020b). This upper bound was originally personally communicated to the authors by Ruosong Wang. We only add it here for completeness. We first give a high level overview of the differences. The proof is almost exactly the same except we now divide the steps of the MDP into \sqrt{H} ‘‘rounds’’. We will brute force search in the rounds for the optimal policy and use the basis constructed in previous work (Du et al., 2020b) to ensure error only grows by a factor of \sqrt{d} . We next prove this in more detail.

Proposition 15 (Ruosong Wang, personal communication) *Assume the MDP has a constant number of actions, optimal value functions V^* and Q^* linear in d dimensional features ψ , time Horizon H and state space at most $\exp(\text{poly}(d))$. Furthermore, assume the feature vectors satisfy $\|\psi(s, a)\|_2 \leq 1$ for all state action pairs and $\|\theta^*\|_2 \leq 1$ for the optimal parameter θ^* . Let $\epsilon \in (0, 1)$. There is an algorithm which takes $\exp(c \cdot \sqrt{H} \log d) / \epsilon^{-2}$ time for some sufficiently large constant c and finds a policy π such that $V^\pi(s) \geq V^*(s) - \epsilon$.*

Proof Given an arbitrary state s , suppose there is a procedure that runs in time

$$T := \exp\left(c \cdot \sqrt{H} \log d\right) / \epsilon^{-2} \cdot \log(1/\delta)$$

and computes an estimation of $\tilde{Q}(s, a)$ for each action $a \in \mathcal{A}$ such that $|\tilde{Q}(s, a) - Q^*(s, a)| \leq \epsilon/(2H)$ with probability at least $1 - \delta$. Then, we claim we can design an algorithm which outputs a policy π such that $V^\pi(s_0) \geq V^*(s_0) - \epsilon$ for the initial state s_0 with probability at least $9/10$. Starting at the state $s = s_0$, we will perform the following steps iteratively:

1. For the current state s , compute the estimations $\tilde{Q}(s, a)$.
2. Choose $a = \arg\max_a \tilde{Q}(s, a)$ and then updates s to be the next state after applying action a .

The above process goes on for at most H iterations. Hence, our estimations $\tilde{Q}(s, a)$ are accurate in all iterations with probability at least $9/10$ if we set $\delta = 1/(10H)$. By Lemma 13, it then holds the

resulting policy is nearly-optimal starting from the initial state s_0 . Moreover, the algorithm runs in time $H \cdot O(T)$, which is within the desired runtime.

To finish the proof, we then describe our procedure for computing the estimations $\{\tilde{Q}(s, a) | a \in \mathcal{A}\}$ for a state s . We will describe the procedure for just the initial state s_0 as computing the estimations for other states can be done similarly. To do so, we divide the time steps of the MDP into \sqrt{H} rounds. For each round $h \in \sqrt{H}$, we build a set of vectors B_h that correspond to the “basis” of some larger set of feature vectors $\psi(s, a)$ where s is a state on the $h \cdot \sqrt{H}$ level, i.e. there is a trajectory going from s_0 to s in $h \cdot \sqrt{H}$ steps. The step is similar to previous work [Du et al. \(2020b\)](#) and proceeds as follows. Let $B_0 = \{\psi(s_0, a) : a \in A\}$ where a is the set of all actions. Then, we construct B_h recursively from $B_{h-1} = \{\psi(s_i, a_i)\}$: Let $\bar{B}_h = \{\psi(P(s_i, a_i), a) : a \in A \text{ and } \psi(s_i, a_i) \in B_{h-1}\}$. Note that $|\bar{B}_h| \leq |A| \cdot |B_{h-1}|$. Next, we set $B_h \subset \bar{B}_h$ as any maximal subset of independent vectors of \bar{B}_h . Note here $|B_h| \leq d$. Moreover, since $\|\psi(s, a)\|_2 \leq 1$ by assumption, any $\psi(s, a) \in \bar{B}_h$ can then be written as $\sum_{i=1}^{|B_h|} \alpha_i \cdot \psi(s_i, a_i)$ satisfying $\|\alpha\|_2 \leq \sqrt{d}$ where $\psi(s_i, a_i)$ are the base vectors in B_h .

We claim that for any $h \in [\sqrt{H}]$, we can learn Q^* on the basis B_h to accuracy $(2d)^{-h}\epsilon$ using at most $O\left(\exp(\sqrt{H} \cdot \log d) \cdot \epsilon^{-2}\right)$ time. We show this via induction on h . Notice that for any state s on the last level (which are 1 step from termination) and action $a \in \mathcal{A}$, the function $Q^*(s, a)$ is simply the expected reward $\mathbb{E}[R(s, a)]$ since the MDP terminates immediately afterwards. Hence, we can follow the same trajectory and sample from $R(s, a)$ for multiple times and compute an empirical mean $\bar{R}(s, a)$. Suppose we take $C \cdot \epsilon^{-2} \log(H \cdot |B_H|/\delta) \cdot (2d)^{\sqrt{H}}$ samples from $R(s, a)$ for a sufficiently large constant C . It then follows from standard concentration inequalities that $|\bar{R}(s, a) - \mathbb{E}[R(s, a)]| \leq (2d)^{-\sqrt{H}}\epsilon$ with probability at least $1 - \delta / (H \cdot |B_H|)$. By the union bound, this holds for all $\phi(s, a) \in B_H$ with probability at least $1 - \delta/H$. Therefore, for all $\phi(s, a) \in B_{\sqrt{H}}$, we can compute an estimator for for $Q^*(s, a)$ with accuracy $(2d)^{-\sqrt{H}}$ in time at most

$$\left(C \cdot \epsilon^{-2} \log(H \cdot |B_H|/\delta) \cdot (2d)^{\sqrt{H}}\right) \cdot |B_H| \cdot \text{poly}(d, H) \leq \exp\left(c \cdot \sqrt{H} \cdot \log d\right) \epsilon^{-2} \log(1/\delta)$$

for some large enough constant c .

Assume we have already learned Q^* on the basis B_h to accuracy $d^{-h}\epsilon$. We will see how we can use the information to estimate Q^* on the basis B_{h-1} to accuracy $d^{-h+1}\epsilon$. Still, consider a single state-action pair (s, a) such that $\phi(s, a) \in B_{h-1}$. Let $\mathcal{R}_{s, a, \sqrt{H}}$ be the set of states reachable from s within \sqrt{H} many steps condition on that the first step is a (notice that $\mathcal{R}_{s, a, \sqrt{H}}$ is a subset of \bar{B}_h by our construction). In other words, each state s' in $\mathcal{R}_{s, a, \sqrt{H}}$ is a state in the $(h \cdot \sqrt{H})$ -th level such that there is a trajectory going from s to s' beginning with the action a . We will without loss of generality assume that each state s' has a unique trajectory starting from s : If there are two different trajectories leading to the same state s' , we can create two copies of s' and index them by the unique trajectory that leads to them.

We know there must exist some state $s^* \in \mathcal{R}_{s, a, \sqrt{H}}$ and $a^* \in \mathcal{A}$ such that $Q^*(s, a)$ is equal to the sum of the expected rewards collected from the trajectory from s to s^* and $Q(s^*, a^*)$. We will denote by $\kappa(s, s')$ the expected reward collected from the path going from s to s' for $s' \in \mathcal{R}_{s, a, \sqrt{H}}$. Our goal is then to compute (i) an estimation for each $\kappa(s, s')$ where $s' \in \mathcal{R}_{s, a, \sqrt{H}}$ and (ii) an estimation for each $Q^*(s', a')$ where $s' \in \mathcal{R}_{s, a, \sqrt{H}}$ and $a' \in A$. It is easy to see if we can compute both (i) and (ii) up to accuracy $d^{-h+1} \cdot 2^{-h}$, we can then take the optimal combination of s', a' to get an estimation of $Q^*(s, a)$ up to accuracy $(2d)^{-h+1}$.

To get an estimation of $\kappa(s, s')$, the expected reward collected from a trajectory, we can just visit the trajectory for multiple times. Since $\kappa(s, s') \leq \sqrt{H}$, it then follow from standard concentration that if we visit the trajectory for

$$C \cdot H \cdot \varepsilon^{-2} \log(H \cdot |\mathcal{R}_{s,a,\sqrt{H}}|/\delta) \cdot (2d)^h$$

many times where C is a sufficiently large constant, then we can estimate all $\kappa(s, s')$ up to the desired accuracy with high probability. There are at most $|\mathcal{A}|^{\sqrt{H}} = \exp(\log |\mathcal{A}| \cdot \sqrt{H})$ many states in $\mathcal{R}_{s,a,\sqrt{H}}$. Since we visit a trajectory $C \cdot H \cdot \varepsilon^{-2} \log(H \cdot |\mathcal{R}_{s,a,\sqrt{H}}|/\delta) \cdot (2d)^h$ times, estimating each $\kappa(s, s')$ takes time at most

$$\begin{aligned} & \exp(\log |\mathcal{A}| \cdot \sqrt{H}) \cdot \left(C \cdot H \cdot \varepsilon^{-2} \log(H \cdot |\mathcal{R}_{s,a,\sqrt{H}}|/\delta) \cdot (2d)^h \right) \cdot \text{poly}(d, H) \\ & \leq \exp\left(c \cdot \sqrt{H} \cdot \log d\right) \varepsilon^{-2} \log(1/\delta) \end{aligned}$$

for some sufficiently large constant c .

To get an estimation of $Q^*(s', a')$ where $s' \in \mathcal{R}_{s,a,\sqrt{H}}$ and $a' \in A$, we will take advantage of the fact that we already have estimations of Q^* on the basis in B_h . In particular, we can express $\psi(s', a') = \sum_{i=1}^{|B_h|} \alpha_i \cdot \psi(s_i, a_i)$ for $\psi(s_i, a_i)$ being the basis in B_h . By linearity, we then have

$$Q^*(s', a') = \sum_{i=1}^{|B_h|} \alpha_i \cdot Q^*(s_i, a_i). \quad (5)$$

On one hand, we have $\|\alpha\|_2 \leq \sqrt{d}$. On the other hand, by the inductive hypothesis, we have an estimation of each $Q^*(s_i, a_i)$ up to accuracy $(2d)^{-h} \cdot \varepsilon$. If we simply plugin our estimation for $Q^*(s_i, a_i)$ into Equation (5) to compute our estimation for $Q^*(s', a')$, we then have the error is at most $d \cdot (2d)^{-h} \cdot \varepsilon \leq 2^{-h} \cdot d^{-h+1} \cdot \varepsilon$ by the Cauchy Schwarz's Inequality, which is the desired bound. Computing the estimation for one $Q^*(s', a')$ takes $\text{poly}(d)$ time. Since there are at most $\exp(\log |\mathcal{A}| \cdot \sqrt{H}) \cdot |\mathcal{A}|$ many pairs of (s', a') , this part takes time at most $\exp\left(c \cdot \left(\sqrt{H} + \log d\right)\right)$ for some sufficiently large constant c .

By induction, this then gives us a way to approximate Q^* on $B_0 = \{\phi(s_0, a) : a \in A\}$ up to accuracy ε with high probability. Moreover, the entire process runs in time $\exp\left(c \cdot \sqrt{H} \cdot \log d\right) \varepsilon^{-2} \log(1/\delta)$ for some sufficiently large constant c . ■

Appendix C. Omitted Proof from Section 4

Proof [Proof of Lemma 6] The first claim follows from the fact that the greedy policy will choose an action that will decrease the distance between the current assignment and the optimal assignment w^* used by the MDP whenever there is such an action. As a result, starting from a state s , it will flip all the free variables where w and w^* differ in the current round, and then flip all the used variables where w and w^* differ in the next round. Upon reaching w^* , the final reward received will be exactly Equation (3) with no intermediate rewards.

Following the greedy policy may fail to reach w^* . The only way this can happen is when the MDP terminates early: when more than $(1 - \epsilon)$ -fraction of the clauses is satisfied, or when we reached a final state in the last round. In such cases, the reward received depends on the extended assignment of the terminal state. From Definitions 3 and 4 it follows that the reward received is the same than the reward would have been if the MDP were not to terminate at that point. Hence, the reward received is still consistent with Equation (3).

To prove the second claim, we follow a similar approach as in the proof of Proposition 10 in Kane et al. (2022). In particular, we will show that $V^\pi(s)$ can be written as a polynomial of degree at most $2p$ in w^* . To see why this is enough, we set θ to be all monomials in w^* of degree at most $2p$. That is, each coordinate of θ corresponds to a multiset $S \subset [v]$ of size $|S| \leq 2p$, and its value is $\theta_S = \prod_{i \in S} w_i^*$. We set $\psi(s)$ to be the corresponding coefficients in the polynomial V^π . Then, we can write $V^\pi(s) = \langle \theta, \psi(s) \rangle$. Since, there are at most $\sum_{i=0}^{2p} v^i \leq 2v^{2p}$ many coefficients we can set the feature dimension as $d = 2v^{2p}$.

Finally, we prove that $V^\pi(s)$ can be written as a polynomial of degree at most $2p$ in w^* . First recall that $\text{dist}_{S, \text{free}}(w, w^*)$ and $\text{dist}_{S, \text{used}}(w, w^*)$ can be written as a linear function of w^* and some state specific parameters $w^{(n)}, w$ and S . Moreover, $\text{dist}(w^{(n)}, w)$ is independent of w^* and only depends on $w^{(n)}$ and w . Then the fact is proven by noting for each $g_i(\cdot)$ in the expression that: (i) for $i < n$ it is independent of w^* ; and (ii) for $i \in \{n, n + 1\}$, it is a degree- p polynomial in $\text{dist}(w^{(n)}, w)$, $\text{dist}_{S, \text{free}}(w, w^*)$ and $\text{dist}_{S, \text{used}}(w, w^*)$.

Finally, note that linear V^π implies linear Q^π in deterministic MDPs for $\psi(s, a) = \psi(P(s, a))$, since by definition, in MDPs with deterministic transition, $Q^\pi(s, a) = r(s, a) + V^\pi(P(s, a))$ and the rewards in our MDPs are zero, except for the last stage where the rewards do not depend on the action. \blacksquare

C.1. Optimality of Greedy Policy

We now prove some structural properties of the polynomials g_i . First, we will show that if a policy makes a “reasonable” number of flips in a round, then the value function decreases by a multiplicative factor. This follows from g_i being (an appropriate degree) Taylor approximation of $\exp(\cdot)$ function around zero.

Claim 16 *The polynomials g_i defined in Equation (2) are bounded:*

$$\frac{1}{4} \leq g_i(x) \leq 1 - \frac{\epsilon}{6bv^{q-2}}$$

for all $\frac{\epsilon}{b} \cdot v \leq x \leq v$ and $i \in \{1, \dots, h\}$. Moreover, g_i is monotonically decreasing.

Proof For simplicity let

$$z = \frac{x}{v^{q-1} \cdot (3 - i/h)}.$$

For the range of values of x we are interested in and since $q \geq 2$, it follows that $z \leq 1/2$. Then, using the fact that g_i is a Taylor approximation, we can upper bound g_i by

$$\begin{aligned} g_i(x) = T_p(-z) &= \sum_{j=0}^p \frac{(-z)^j}{j!} \leq 1 - \frac{z}{2} && \text{(as } p \geq 2 \text{ and } z \leq 1/2) \\ &\leq 1 - \frac{\epsilon}{6bv^{q-2}}. && \text{(as } x \geq \frac{\epsilon}{b} \cdot v) \end{aligned}$$

On the other hand, we can lower bound $g_i(x)$ as follows:

$$g_i(x) = \exp(-z) - \sum_{j=p+1}^{\infty} \frac{(-z)^j}{j!} \geq \exp\left(-\frac{1}{2}\right) - \frac{1}{2^p(p+1)!} \geq \frac{1}{4}, \quad (6)$$

where the first inequality again follows from $z \leq 1/2$ and summation of geometric series and the last inequality holds as long as $p \geq 1$.

Next, we argue $g_i(x)$ is monotonically decreasing. We do so by showing the derivative of $g_i(x)$ is negative. For this, we calculate

$$\frac{d}{dx}g_i(x) = \left(\frac{d}{dx}z\right) \cdot \left(\frac{d}{dz}T_p(-z)\right) = -\sum_{j=0}^{p-1} \left(\frac{(-z)^j}{j!}\right).$$

Similar to Equation (6), we have

$$\sum_{j=0}^{p-1} \left(\frac{(-z)^j}{j!}\right) \geq \exp\left(-\frac{1}{2}\right) - \frac{1}{2^{p-1}p!} > 0$$

whenever $p \geq 2$. Therefore, $\frac{d}{dx}g_i(x) < 0$ which implies that $g_i(x)$ is monotonically decreasing. ■

Next, we will show that the polynomials are designed such that correcting variables (where w and w^* differ) in round i is always better than correcting variables in round $i+1$. In particular, suppose we have flipped c bits in the i -th round and d bits in the $(i+1)$ -th round. We then want to show that $g_i(c) \cdot g_{i+1}(d) \geq g_i(c-1) \cdot g_{i+1}(d+1)$ for any $1 \leq c \leq v$ and $0 \leq d \leq v$. To prove this, we need to show that the error from Taylor approximation which depends on the choice of p is relatively small.

Claim 17 For any two polynomials g_i, g_{i+1} defined in Equation (2), let

$$f_{i,c,d}(x) = g_i(c+x) \cdot g_{i+1}(d-x).$$

where $i \in \{1, \dots, h\}$, $0 \leq c, d \leq v$ and $x \in \{1, 2, \dots, d\}$. Then, for large enough v ,

$$f_{i,c,d}(x) \geq f_{i,c,d}(x-1).$$

Proof Consider the function $\hat{f}_{i,c,d}(\cdot)$ defined as

$$\hat{f}_{i,c,d}(x) = \exp\left(-\frac{c+x}{v^{q-1} \cdot (3-i/h)}\right) \cdot \exp\left(-\frac{d-x}{v^{q-1} \cdot (3-(i+1)/h)}\right). \quad (7)$$

To prove our claim, we will show that

$$\hat{f}_{i,c,d}(x) - \hat{f}_{i,c,d}(x-1) \geq \Omega\left(\frac{1}{\alpha \cdot v^{2q-2}}\right), \quad (8)$$

$$|f_{i,c,d}(x) - \hat{f}_{i,c,d}(x)| = O\left(\frac{1}{v^{2q-2}}\right), \quad (9)$$

where to recall α in Equation (8) is the parameter in the time horizon factor, i.e. $H = \alpha \cdot v^q$ and $h = H/v$. Then, our claim follows from the inequalities above as long as α is set to be a sufficiently small constant. We first prove Equation (8). For this, we will show that the derivative of $\hat{f}_{i,c,d}(x)$ is not only positive but lower bounded by $\Omega(1/\alpha \cdot v^{2q-2})$. The derivative of $\hat{f}(\cdot)$ is given by

$$\hat{f}'_{i,c,d}(x) = \hat{f}_{i,c,d}(x) \cdot \frac{h}{v^{q-1} \cdot (3h-i) \cdot (3h-i-1)}. \quad (10)$$

Notice that we always have

$$\hat{f}_{i,c,d}(x) \geq \exp\left(-\frac{2v}{v^{q-1}}\right) \cdot \exp\left(-\frac{v}{v^{q-1}}\right) = \exp\left(-\frac{3}{v^{q-2}}\right) \geq \Omega(1),$$

where the first step follows from $0 \leq c, d, x \leq v$ and $i \in \{1, \dots, h\}$ and last step from $q \geq 2$. We hence have $\hat{f}_{i,c,d}(x) \geq \Omega(1)$. Combining this with $h := \alpha \cdot v^{q-1}$ and Equation (10), we can lower bound the derivatives by

$$\hat{f}'_{i,c,d}(x) \geq \Omega\left(\frac{1}{\alpha \cdot v^{2q-2}}\right).$$

Since $\hat{f}_{i,c,d}$ is a convex function, this proves Equation (8).

Next, we prove Equation (9). Recall that

$$g_i(y) = T_p\left(\frac{-y}{v^{q-1} \cdot (3-i/h)}\right)$$

where T_p is the degree- p Taylor approximation of the exponential function. Then, for $0 \leq y \leq 2v$ we have

$$\left|g_i(y) - \exp\left(-\frac{y}{v^{q-1} \cdot (3-i/h)}\right)\right| \leq O\left(\left(\frac{y}{v^{q-1} \cdot (3-i/h)}\right)^{p+1}\right) \leq O\left(\frac{1}{v^{(q-2) \cdot (p+1)} \cdot 2^{p+1}}\right). \quad (11)$$

In addition, for $y \geq 0$ we have

$$\exp(-y/(v^{q-1} \cdot (3-i/h))) < 1. \quad (12)$$

Substituting Equations (11) and (12) into Equation (7) then gives

$$\begin{aligned} f_{i,c,d}(x) &= \left(\exp\left(-\frac{c+x}{v^{q-1} \cdot (3-i/h)}\right) \pm O\left(\frac{1}{v^{(q-2) \cdot (p+1)} \cdot 2^{p+1}}\right)\right) \cdot \\ &\quad \left(\exp\left(-\frac{d-x}{v^{q-1} \cdot (3-(i+1)/h)}\right) \pm O\left(\frac{1}{v^{(q-2) \cdot (p+1)} \cdot 2^{p+1}}\right)\right) \\ &= \hat{f}_{i,c,d}(x) \pm O\left(\frac{1}{v^{(q-2) \cdot (p+1)} \cdot 2^{p+1}}\right). \end{aligned}$$

For both settings of p and q we consider, $p = 2$; $q = 4$ for the first result or $p = 2 \log v$; $q = 2$ for the second result, this implies

$$|f_{i,c,d}(x) - \hat{f}_{i,c,d}(x)| \leq O\left(\frac{1}{v^{(q-2) \cdot (p+1)} \cdot 2^{p+1}}\right) = O\left(\frac{1}{v^{2q-2}}\right).$$

■

Using above recursively, we can show that any greedy policy is an optimal policy in our MDPs. This is important because this in conjunction with Lemma 6 implies that the optimal value functions V^* and Q^* can be written as a linear function of some features depending only on states s and action a .

Lemma 18 *Any greedy policy as defined in Definition 5 is optimal.*

Proof Let π be the greedy policy and consider some other policy $\tilde{\pi}$. We show the reward received by the greedy policy π is no worse than $\tilde{\pi}$ starting from an arbitrary state s_{curr} with assignment w_{curr} , free variables S_{curr} and round history $w^{(1)}, \dots, w^{(n)}$. Notice that the final rewards of both $\tilde{\pi}$ and π will have the term $G := \prod_{i=1}^{n-1} g_i(\text{dist}(w^{(i)}, w^{(i+1)}))$. We abbreviate the term as G so that we can focus on comparing the remaining terms.

Suppose $\tilde{\pi}$ terminates in the $(n+l)$ -th round. In particular, assume it terminates on the state \tilde{s} with round history $w^{(1)}, \dots, w^{(n)}, \tilde{w}^{(n+1)}, \dots, \tilde{w}^{(n+l)}$, free variables \tilde{S} and terminal assignment \tilde{w} . For notational convenience, we will denote $\tilde{w}^{(n+l+1)} = \text{ext}(\tilde{w}, \tilde{S})$.⁴ Then, the value of $\tilde{\pi}$ starting from w_{curr} (also the reward of the state \tilde{s}) can be written as

$$V^{\tilde{\pi}}(s_{\text{curr}}) = G \cdot \prod_{i=n}^{n+l} g_i \left(\text{dist}(\tilde{w}^{(i)}, \tilde{w}^{(i+1)}) \right) \cdot g_{n+l+1} \left(\text{dist}(\tilde{w}^{(n+l+1)}, w^*) \right). \quad (13)$$

First, we argue that it is never beneficial for $\tilde{\pi}$ to terminate in rounds after the $(n+1)$ -th round. More formally, we will show

$$\prod_{i=n}^{n+l} g_i(\text{dist}(\tilde{w}^{(i)}, \tilde{w}^{(i+1)})) \cdot g_{n+l+1}(\text{dist}(\tilde{w}^{(n+l+1)}, w^*)) \leq g_n(\tilde{w}^{(n)}, \tilde{w}^{(n+1)}) \cdot g_{n+1}(\tilde{w}^{(n+1)}, w^*). \quad (14)$$

Using Claim 17, we have

$$\begin{aligned} & g_{n+l} \left(\text{dist}(\tilde{w}^{(n+l)}, \tilde{w}^{(n+l+1)}) \right) \cdot g_{n+l+1} \left(\text{dist}(\tilde{w}^{(n+l+1)}, w^*) \right) \\ & \leq g_{n+l} \left(\text{dist}(\tilde{w}^{(n+l)}, \tilde{w}^{(n+l+1)}) + \text{dist}(\tilde{w}^{(n+l+1)}, w^*) \right) \cdot g_{n+l+1}(0) \leq g_{n+l} \left(\text{dist}(\tilde{w}^{(n+l)}, w^*) \right), \end{aligned}$$

where the last inequality follows from $g_{n+l+1}(0) = 1$, the triangle inequality used with $\text{dist}(\cdot)$ and that $g_{n+l}(\cdot)$ is monotonically decreasing (Claim 16). This then shows that

$$\begin{aligned} & \prod_{i=n}^{n+l} g_i(\text{dist}(\tilde{w}^{(i)}, \tilde{w}^{(i+1)})) \cdot g_{n+l+1}(\text{dist}(\tilde{w}^{(n+l+1)}, w^*)) \\ & \leq \prod_{i=n}^{n+l-1} g_i(\text{dist}(\tilde{w}^{(i)}, \tilde{w}^{(i+1)})) \cdot g_{n+l}(\text{dist}(\tilde{w}^{(n+l)}, w^*)). \end{aligned}$$

4. Notice it could be that the terminal state \tilde{s} is in the same round as s_{curr} . In that case, we have $l = 0$.

We can then do induction on l to get Equation (14). Substituting Equation (14) into Equation (13) then gives

$$V^{\tilde{\pi}}(s) \leq G \cdot g_n \left(\text{dist} \left(w^{(n)}, \tilde{w}^{(n+1)} \right) \right) \cdot g_{n+1} \left(\text{dist} \left(\tilde{w}^{(n+1)}, w^* \right) \right). \quad (15)$$

We then proceed to argue the expression above is upper bounded by V^π . Notice that by the triangle inequality, for any $\tilde{w}^{(n+1)}$ it holds that

$$\text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)}) + \text{dist}(\tilde{w}^{(n+1)}, w^*) \geq \text{dist}(w_{\text{curr}}, w^*).$$

On the other hand, we always have

$$\text{dist}(w_{\text{curr}}, w^*) = \text{dist}_{S_{\text{curr,used}}}(w_{\text{curr}}, w^*) + \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*).$$

Combining the two and rearranging the terms then gives

$$\text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)}) + \text{dist}(\tilde{w}^{(n+1)}, w^*) - \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*) \geq \text{dist}_{S_{\text{curr,used}}}(w_{\text{curr}}, w^*) \quad (16)$$

Now, we will use case analysis based on the relative sizes of $\text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)})$ and $\text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*)$. We first consider the case $\text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)}) \leq \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*)$. In this case we have

$$\begin{aligned} & g_n \left(\text{dist}(w^{(n)}, \tilde{w}^{(n+1)}) \right) \cdot g_{n+1} \left(\text{dist}(\tilde{w}^{(n+1)}, w^*) \right) \\ &= g_n \left(\text{dist}(w^{(n)}, w_{\text{curr}}) + \text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)}) \right) \cdot g_{n+1} \left(\text{dist}(\tilde{w}^{(n+1)}, w^*) \right) \\ &\leq g_n \left(\text{dist}(w^{(n)}, w_{\text{curr}}) + \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*) \right) \\ &\cdot g_{n+1} \left(\text{dist}(\tilde{w}^{(n+1)}, w^*) + \text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)}) - \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*) \right) \\ &\leq g_n \left(\text{dist}(w^{(n)}, w_{\text{curr}}) + \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*) \right) \cdot g_{n+1} \left(\text{dist}_{S_{\text{curr,used}}}(w_{\text{curr}}, w^*) \right), \end{aligned}$$

where the first inequality follows from Claim 17 and the second inequality follows from g_n is a monotonically decreasing function (Claim 16) and Equation (16).

Now, we consider the other remaining case when $\text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)}) > \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*)$. Denote M as the set of free variables on which w_{curr} and w^* agree but w_{curr} and $\tilde{w}^{(n+1)}$ disagree. In other words, these are the variables mistakenly flipped by the policy $\tilde{\pi}$ on the path from w_{curr} to $\tilde{w}^{(n+1)}$. Since $\tilde{w}^{(n+1)}, w^*$ disagree on these variables, these variables must be flipped again on the path from $\tilde{w}^{(n+1)}$ to w^* . We can then consider the alternative path $w_{\text{curr}} \rightarrow \bar{w} \rightarrow w^*$ for \bar{w} satisfying $\bar{w}_i \neq \tilde{w}_i^{(n+1)}$ for $i \in M$ and $\bar{w}_i = \tilde{w}_i^{(n+1)}$ for $i \notin M$. Then, it is easy to see that

$$\begin{aligned} & g_n(\text{dist}(w^{(n)}, w_{\text{curr}}) + \text{dist}(w_{\text{curr}}, \tilde{w}^{(n+1)})) \cdot g_{n+1}(\text{dist}(\tilde{w}^{(n+1)}, w^*)) \\ &\leq g_n(\text{dist}(w^{(n)}, w_{\text{curr}}) + \text{dist}(w_{\text{curr}}, \bar{w})) \cdot g_{n+1}(\text{dist}(\bar{w}, w^*)) \end{aligned}$$

since $g_n(\cdot)$ is monotonically decreasing. Moreover, now we have $\text{dist}(w_{\text{curr}}, \bar{w}) \leq \text{dist}_{S_{\text{curr,free}}}(w_{\text{curr}}, w^*)$ since the variables flipped are restricted to be the ones on which w_{curr} and w^* do not agree. Hence, the proof is reduced to the first case. ■

C.2. Reduction from GAP-3-SAT to LINEAR-k-RL

Proof [Proof of Proposition 8] Take a satisfiable formula φ . The optimal value in this case is at least $1/4$. Indeed, by Lemma 18, the greedy policy is optimal, its value is $g_1(\text{dist}(w, w^*))$ and thus by Claim 16,

$$V^* = g_1(\text{dist}(w, w^*)) \geq \frac{1}{4}.$$

We now argue by contraposition: Assume that π does not end on an assignment that satisfies at least a $(1 - \epsilon)$ -fraction of clauses. Let $w^{(1)}, \dots, w^{(h)}, \bar{w}$ denote the sequence of assignments obtained by π : $w^{(1)} = w$, and $w^{(i+1)}$ is the assignment at the end of round $1 \leq i \leq h - 1$ and \bar{w} is the final assignment. Recall in each round the MDP has two stages. In the first stage, the agent is presented unsatisfied clauses made up of only free variables. By our construction, the first stage is of length at least $\epsilon v/b$. It follows that $\text{dist}(w^{(i)}, w^{(i+1)}) \geq \epsilon v/b$ since the policies are not allowed to undo any flips. We can then upper bound the reward obtained at the end by

$$\prod_{i=1}^h g_i(\text{dist}(w^{(i)}, w^{(i+1)})) \cdot g_{h+1}(\text{dist}(\bar{w}, w^*)) \leq \prod_{i=1}^h g_i(\epsilon v/b) \leq \left(1 - \frac{\epsilon}{6bv^{q-2}}\right)^h \leq \exp(-cv),$$

where $c = \Theta(\alpha\epsilon/b)$, the first inequality follows $\text{dist}(w^{(i)}, w^{(i+1)}) \geq \epsilon v/b$, the second from Claim 16, and the third follows from $1 - x \leq e^{-x}$ that holds for all x and our choice of $h = \alpha v^{q-1}$. Therefore, if $V^\pi > V^* - 1/8 \geq 1/8$, and v is large enough so that $\exp(-cv) < 1/8$, then the policy π has to end on a state which satisfied at least a $(1 - \epsilon)$ -fraction of clauses. ■

Proof [Proof of Proposition 9]

Let \Pr_{M_φ} and $\Pr_{\bar{M}_\varphi}$ denote the distribution on the observed rewards and output policies induced by the algorithm \mathcal{A}_{RL} when running on access to MDP oracle M_φ and simulator \bar{M}_φ respectively. Let R_i denote the reward received on the last layer at the end of i -th trajectory and N be the total number of trajectories sampled by algorithm \mathcal{A}_{RL} when running on access to MDP oracle M_φ . By our assumption, \mathcal{A}_{RL} runs in time T and therefore $N \leq T$.

We remark that if the algorithm \mathcal{A}_{RL} ever reaches a satisfying assignment, \mathcal{A}_{SAT} will terminate the simulation immediately, returning YES. Before reaching a satisfying assignment, \mathcal{A}_{RL} may only receive rewards from the last layer. Since the expected reward at the last layer in the MDP M_φ is upper bounded by $1/(5T)$ by our assumption, and the algorithm only visits at most $N \leq T$ states on last layer, we get by the union bound that with high probability all the rewards at the last level are zero. More precisely, we have

$$\Pr_{M_\varphi} [R_i = 0 \forall i \in [N]] \geq 1 - T/(5T) \geq \frac{4}{5}.$$

We say that \mathcal{A}_{RL} succeeds with access to M_φ (or \bar{M}_φ) if the output policy π satisfies $V^\pi > V^* - 1/8$ with respect to M_φ after running for time at most T . Using the above reasoning and the assumption that \mathcal{A}_{RL} succeeds with access to MDP oracle M_φ with probability $9/10$ implies

$$\Pr_{M_\varphi} [\mathcal{A}_{RL} \text{ succeeds with access to } M_\varphi \mid R_i = 0 \forall i \in [N]] \geq \frac{\frac{9}{10} - \frac{1}{5}}{\frac{4}{5}} = \frac{7}{8}.$$

Note that the marginal distributions \Pr_{M_φ} and $\Pr_{\bar{M}_\varphi}$ conditioned on $R_i = 0 \forall i \in [N]$ are exactly the same because MDP oracle \bar{M}_φ and simulator M_φ may only differ on last layer rewards before \mathcal{A}_{RL} reaches a satisfying assignment. This implies

$$\begin{aligned} & \Pr_{\bar{M}_\varphi} [\mathcal{A}_{RL} \text{ succeeds with access to } \bar{M}_\varphi \mid R_i = 0 \forall i \in [N]] \\ &= \Pr_{M_\varphi} [\mathcal{A}_{RL} \text{ succeeds with access to } M_\varphi \mid R_i = 0 \forall i \in [N]] \end{aligned}$$

Since, $\Pr_{\bar{M}_\varphi} [R_i = 0 \forall i \in [N]] = 1$, we conclude that

$$\Pr_{\bar{M}_\varphi} [\mathcal{A}_{RL} \text{ succeeds with access to } \bar{M}_\varphi] \geq \frac{7}{8}.$$

■