No-Regret Learning in Dynamic Stackelberg Games

Niklas Lauffer, Mahsa Ghasemi, Abolfazl Hashemi, Member, IEEE, Yagiz Savas, and Ufuk Topcu

Abstract—In a Stackelberg game, a leader commits to a randomized strategy and a follower chooses their best strategy in response. We consider an extension of a standard Stackelberg game, called a discrete-time dynamic Stackelberg game, that has an underlying state space that affects the leader’s rewards and available strategies and evolves in a Markovian manner depending on both the leader and follower’s selected strategies. Although standard Stackelberg games have been utilized to improve scheduling in security domains, their deployment is often limited by requiring complete information of the follower’s utility function. In contrast, we consider scenarios where the follower’s utility function is unknown to the leader; however, it can be linearly parameterized. Our objective is then to provide an algorithm that prescribes a randomized strategy to the leader at each step of the game based on observations of how the follower responded in previous steps. We design an online learning algorithm that, with high probability, is no-regret, i.e., achieves a regret bound (when compared to the best policy in hindsight), which is sublinear in the number of time steps; the degree of sublinearity depends on the number of features representing the follower’s utility function. The regret of the proposed learning algorithm is independent of the size of the state space and polynomial in the rest of the parameters of the game. We show that the proposed learning algorithm outperforms existing model-free reinforcement learning approaches.

Index Terms—Iterative learning control, no-regret learning, online learning, Stackelberg games, stochastic games.

I. INTRODUCTION

STACKELBERG games model strategic interactions between two agents: a leader and a follower [1], [2]. The leader plays first by committing to a randomized strategy. The follower observes the leader’s commitment and then plays a strategy to best respond to the leader’s chosen strategy.

Stackelberg games have been successfully applied to a multitude of scenarios to model competition between firms [1], improve security scheduling [3], and dictate resource allocation [4]. The Los Angeles Airport and other security agencies have deployed randomized patrol routes based on Stackelberg models [5]. Researchers have also used Stackelberg models to improve national park wildlife ranger patrol patterns and resource distribution to protect against illegal poaching [6].

Repeated Stackelberg games [7] are used to model agents that repeatedly interact in a Stackelberg game. A significant drawback of standard repeated Stackelberg game formulations is their inability to model dynamic scenarios. In scenarios involving repeated interaction between two strategic agents, decisions made early on can sometimes have long-lasting effects. Moreover, repeated Stackelberg formulations cannot model dynamic changes in the leader’s preferences or available strategies over time.

Consider a scenario in which park rangers are responsible for protecting a geographical area containing different types of animals from illegal poaching. The geographical area is split into distinct regions, each of which contains a different density of the animals. Each month, the park rangers can deploy a mixed strategy to decide which of the regions to patrol. After observing the rangers’ strategy, the poachers can attempt to lay snares in any one of the regions. If the poachers attempt to lay snares in a region being patrolled by the park rangers, they are caught and penalized. Otherwise, the snare has a chance of catching one of the animals in that region based on the density of each of the animals.

The park ranger’s resources are limited. Over the course of each year, the park rangers are allocated a budget for anti-poaching patrols. The further a region is from the park headquarters, the more expensive it is to patrol. If the park rangers deplete their budget, they are unable to launch any more patrols until the following year. Moreover, the damage that poaching incurs on an animal population can fluctuate throughout the year. For example, the mating season for rhinos typically occurs in the months of October and November, during which poaching...
is especially damaging to the rhino population. Such a dynamic time-varying setting requires a model with an underlying state space.

In order to address the limitations of existing formulations, we introduce discrete-time dynamic Stackelberg games (DSGs), an extension of the standard repeated Stackelberg formulation that includes an underlying persistent discrete-time state space. The state space allows DSGs to model scenarios with dynamic utility functions, resources, and other states. The state can vary across steps of the DSG in a Markovian manner dependent on the actions of both the leader and the follower. In a DSG, the state is only relevant to the leader. The leader’s reward and the set of available actions are directly dependent on the current state, but the follower’s are not.

Modeling a scenario as a standard Stackelberg game (and thus also a DSG) requires complete knowledge of the follower’s payoff in different scenarios. In practice, the parameters of such models are often estimated based on historical data [8]. However, small modeling imperfections can lead to significant inefficiencies in the utility of the strategies derived from solving the Stackelberg game. Arguably, the most severe modeling limitation is that in order to compute the optimal strategy, the leader must precisely know the follower’s utility function.

We study the online learning problem of sequentially synthesizing policies for the leader in a DSG under a setting in which the follower’s utility function is unknown. In this setting, the leader must interact with the follower and incrementally learn its behavior over time by using past interactions with the follower to improve future decisions.

A. Contributions

We introduce a new modeling formalism, called a discrete-time DSG, that bridges the modeling formalism of repeated Stackelberg games and Markov decision processes (MDPs). We propose an online learning algorithm for computing an adaptive policy for the leader in a scenario in which the follower’s reward function is unknown. The proposed algorithm enables the leader to create an estimate of the follower’s utility function and use that estimate to update its policy. We prove that the proposed algorithm achieves a sublinear regret bound with respect to the time horizon — establishing the first no-regret online learning algorithm in this setting. The regret bound’s degree of sublinearity depends on the number of features representing the follower’s utility function. It is independent of the size of the state space and polynomial in the rest of the parameters of the game. Through a series of experiments, we evaluate the empirical performance of our algorithm and demonstrate the practical application of our approach.

B. Related Work

Generating policies for repeated Stackelberg games in a setting where the follower’s utility function is unknown to the leader is an active area of research. In this setting, the agents play in some variation of a repeated Stackelberg game where the leader makes decisions based on observations from previous steps in the game. The authors in [9] and [10] study this problem with the primary objective of designing an algorithm that learns the follower’s utility with low sample complexity, i.e., in as few repetitions of the game as possible. Then, the learned representation can be used to play near optimally for the rest of time. Other variations of this problem include an online setting in [11], where the follower’s utility function can change (potentially adversarially) over time and the objective is to design a no-regret learning algorithm, that is, an algorithm that asymptotically converges to the optimal strategy. However, none of these works consider a dynamic scenario. Since the follower’s utility function in a DSG is independent of the state space, it is natural to desire a regret bound independent of the size of the state space. This precludes the use of existing online learning algorithms for repeated Stackelberg games since an independent game would have to be learned and solved for each state in the state space.

DSGs are also closely related to stochastic games, a well-studied class of game played over a state space where payoffs and transitions are also determined by the actions from a pair of agents [12]. Online learning in stochastic games is also an ongoing area of research [13]. Closely related to our work, Ouyang et al. [14] give a general solution concept for stochastic dynamic games with asymmetric or unknown information. In contrast to DSGs, agents in a stochastic game choose actions simultaneously such that neither agent can observe its opponent’s action before selecting its own action.

DSGs are similar to existing feedback Stackelberg games [15] in which two agents play in a Stackelberg game over a state space in which individual policies are determined for each step of the game. Chen and Cruz [16] introduce a solution concept for feedback Stackelberg games in which the leader knows both utility functions and the follower only knows their own utility without considering any learning. Feedback Stackelberg games are typically studied in continuous settings modeled by differential equations with perfect information, that is, policies are computed with full knowledge of the follower’s utility function. In contrast, we study a setting in which the follower’s utility function is unknown to the leader.

MDPs are commonly used to model a single decision-making agent interacting with an environment [17]. Online learning in MDPs is an active area of research. Neu et al. [18] give an online learning algorithm in an MDP, where the rewards are unknown and chosen by an adversary, but the transitions are known. By disregarding the reward and transition structure induced by the Stackelberg game, a DSG can be reduced to an MDP with an unknown reward and transition function. This makes it possible to treat the learning problem in a DSG as a reinforcement learning problem in an MDP. However, reinforcement learning algorithms fail to incorporate the reward and transition structure inherent in the DSG; their learning rate scales with the size of the state space. Moreover, if the leader in the DSG has action space $\mathcal{A}$, the resulting MDP after the transformation has a continuous action space in $\mathbb{R}^{|\mathcal{A}|}$, making this approach intractable even for small $|\mathcal{A}|$. In Section V, we directly compare our learning algorithm against a model-free reinforcement learning approach.
Parameters: Discrete-time dynamic Stackelberg game \( \langle S, A, B, r, u, P \rangle \) and time horizon \( T \).

For all episodes \( t = 1, 2, \ldots, T \), repeat

For all steps \( h = 1, 2, \ldots, H \), repeat

1) The learner observes the current state \( s \in S_h \).
2) Based on previous observations, the learner selects mixed strategy \( x \in \Delta(A(s)) \).
3) The follower responds with action \( b \in B \) that maximizes its expected utility \( E_{\pi \sim x} [u(a, b)] \).
4) An action \( a \sim x \) is sampled.
5) The learner obtains reward \( r(s, a, b) \) and the environment transitions to the next state \( s' \sim P(\cdot | s, a, b) \).

Fig. 1. Protocol for online learning in a discrete-time dynamic Stackelberg game.

C. Organization

After giving a formal construction of DSGs and a formal description of the learning problem in Section II, we introduce a novel algorithm in Section III based on optimistically choosing policies that are consistent with previous observations. In Section IV, we analyze the algorithm and show that it achieves a regret that, with high probability, is sublinear in the number of time-steps. In Section V, we experimentally demonstrate the impact of varying parameters of the game on regret and compare against a model-free reinforcement learning approach. Finally, Section VI concludes this article.

II. Problem Definition

A discrete-time DSG is played between a leader agent and a follower agent. The game is played sequentially on a six-tuple \( \langle S, A, B, r, u, P \rangle \), whose elements are defined as follows.

1) \( S \) is a set of states.
2) \( A = \{a_1, \ldots, a_m\} \) is a set of actions available to the leader. We denote the leader’s available actions at a state \( s \in S \) by \( A(s) \subseteq A \).
3) \( B = \{b_1, \ldots, b_m\} \) is a set of actions available to the follower.
4) \( r : S \times A \times B \rightarrow \mathbb{R} \) is the reward function for the leader.
5) \( u : A \times B \rightarrow \mathbb{R} \) is the utility function for the follower. (Notice the lack of dependency on the state space \( S \).)
6) \( P : S \times A \times B \times S \rightarrow [0, 1] \) is the transition function that satisfies \( \sum_{s' \in S} P(s, a, b, s') = 1 \).

We outline the interactions between the leader and the follower in a DSG in Fig. 1. First, the leader chooses a mixed strategy \( x_\pi \in \Delta(A(s)) \) in state \( s \in S \), where \( \Delta \) is the probability simplex of appropriate dimension. Then, the follower chooses an action \( b \in B \) in response such that

\[
b \in \arg \max_{b \in B} E_{a \sim x_\pi} [u(a, b)].
\]

Note that in this setting, the follower is myopic and aims to maximize its immediate expected utility, a standard assumption made in Stackelberg learning problems [7], [9], [10], [11].

An action \( a \sim x_\pi \) is sampled from the leader’s mixed strategy to determine the next state. We limit the scope of this article to the class of episodic DSGs. That is, we assume that the state space \( S = S_1 \cup \cdots \cup S_H \) is divided into \( H \) disjoint layers with \( S_1 = \{s_1\} \), and the game is separated into a series of \( T \) episodes. We denote by \( s_{t,h} \in S_h \) the state occupied in the \( h \)th step of episode \( t \). Then, at a state \( s_{t,h} \in S_h \), the next state \( s_{t,h+1} \in S_{h+1} \) is guaranteed to be contained within the next layer, i.e., \( \sum_{s_{t,h+1} \in S_{h+1}} P(s_{t,h+1} | s_{t,h}, a, b, s_{t,h+1}) = 1 \).

Suppose that, at a state \( s_{t,h} \), the leader takes the action \( a_{t,h} \) and the follower takes the action \( b_{t,h} \) in response. Then, the leader receives the reward \( r_{t,h} = r(s_{t,h}, a_{t,h}, b_{t,h}) \) for step \( h \) of the episode. Consequently, the leader’s cumulative reward for the episode becomes \( \sum_{h=1}^H r_{t,h} \).

Our objective in this article is to design an algorithm which the leader can use to learn a strategy that maximizes its expected cumulative reward over all episodes. To represent the leader’s expected reward compactly, we can think of the follower’s response as a function \( \varphi : \Delta(A) \rightarrow \mathbb{R} \), where

\[
\varphi(x) = \arg \max_{b \in B} E_{a \sim x} [u(a, b)].
\]

Then, the leader’s reward function over actions induces an auxiliary reward function \( R : S \times \Delta(A) \rightarrow \mathbb{R} \) over mixed strategies given by

\[
R(s, x) = E_{a \sim x} [r(s, a, \varphi(x))].
\]

Let \( \pi : S \rightarrow \Delta(A) \) be a policy that describes the leader’s action selection for each state in the DSG. We denote by \( \Pi \) the set of all policies \( \pi \). The optimal policy for a leader that knows the follower’s utility function \( u \) is given by

\[
\pi^* \in \arg \sup_{\pi \in \Pi} E^\pi \left[ \sum_{t=1}^T \sum_{h=1}^H R(s_{t,h}, \pi(s_{t,h})) \right]
\]

where the expectation is taken over the trajectories (represented by \( s_{t,h} \)) induced by the leader’s policy \( \pi \), the follower’s corresponding response as defined in (1), and the stochasticity in the transition function of the DSG. In this article, we assume that the follower’s utility function is unknown to the leader. Hence, the leader needs to learn a sequence of policies through interactions with the follower.

We use regret [19] to evaluate the asymptotic performance of a learning algorithm. Regret measures the difference between the cumulative reward of a learning agent and the cumulative reward of the best strategy in hindsight. In our case, the best strategy in hindsight is the best strategy given that the follower’s utility function was known to the leader ahead of time, i.e., the policy in (4).

Definition 1 (Regret): The regret \( R_T \) of a sequence of policies \( \{\pi_{t,h} \in \Pi : t \in [T], h \in [H]\} \) is given by

\[
\mathbb{E}[R_T] = \sum_{t=1}^T \sum_{h=1}^H R(s_{t,h}, \pi^*(s_{t,h})) - \mathbb{E} \left[ \sum_{t=1}^T \sum_{h=1}^H R(s_{t,h}, \pi_{t,h}(s_{t,h})) \right].
\]
The main problem investigated in this article is as follows.

**Problem 1:** Suppose that the follower’s utility function $u$ is fixed and unknown to the leader. Provide an online learning algorithm that computes a sequence of policies that minimizes the leader’s regret $R_T$.

### A. Assumptions

Before proceeding with the learning algorithm that solves Problem 1, we list our assumptions on the interactions between the leader and the follower.

**Assumption 1:** The range of the leader’s reward function is $[0, 1] \subset \mathbb{R}$, i.e., $r(s, a, b) \in [0, 1]$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $b \in \mathcal{B}$.

We introduce the above assumption for notational simplicity. Note that if the range of the reward function instead falls in some other closed interval $[a, b]$, rewards can be normalized, without loss of generality, to lie in the interval $[0, 1]$.

We adopt the solution concept of **strong Stackelberg equilibrium** that ties are broken in the favor of the leader.

**Assumption 2 (Strong Stackelberg equilibrium):** Let $\tau(x) \subset \mathcal{B}$ represent the set of the follower’s best responses to a leader’s mixed strategy $x \in \Delta(\mathcal{A})$. In any state $s \in \mathcal{S}$, if the leader plays a mixed strategy $x$, the follower is guaranteed to play an action $b \in \tau(x)$ such that, for all $b' \in \tau(x)$

$$
\mathbb{E}_{a \sim \pi}[r(s, a, b)] \geq \mathbb{E}_{a \sim \pi}[r(s, a, b')].
$$

Strong Stackelberg equilibrium is the most common solution concept for Stackelberg games since it ensures the existence of an optimal strategy [20]. It also allows some algorithmic simplification in defining the feasibility region of optimization problems that solve for optimal leader policies.

**Assumption 3 (Linear function approximation):** The follower’s utility function is linearly parameterized. That is, there exists a feature mapping $f: \mathcal{A} \times \mathcal{B} \rightarrow \mathbb{R}^p$ to the leader such that, for some $\theta^* \in \mathbb{R}^p$:

$$
u(a, b) = (f(a, b), \theta^*).$$

Linear function approximation is commonly used in reinforcement learning [21], [22] and online learning [23].

Finally, for ease of notation, we define the following matrices, which we call *feature matrices*, for each action $b \in \mathcal{B}$. $\mathbf{M}_b \in \mathbb{R}^{n \times p}$ such that the $i$th row of $\mathbf{M}_b$ is $f(a_i, b)$.

### B. Best Strategy in Hindsight

Before discussing our algorithm for the general case, we first investigate the case in which the follower’s utility function is known to gain some intuition.

If the follower’s utility function is known, the leader’s optimal policy $\pi^* : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ is the solution to a bilevel optimization problem. By extending the linear program (LP) used in [24] to solve a standard Stackelberg game, the optimal mixed strategy determined by policy $\pi^*$ in state $s$ is given by the solution to the following optimization problem:

$$
\max_{\pi_s} V(\pi^*, s)
$$

s.t. $V(\pi^*, s) = \mathbb{E}_{a \sim \pi} \left[r(s, a, b) + \sum_{s'} P(s, a, b, s') V(\pi^*, s')\right]$ \hspace{1cm} (8b)

$x_T^T (\mathbf{M}_b - \mathbf{M}_{b'}) \theta^* \geq 0 \ \forall b' \in \mathcal{B}$ \hspace{1cm} (8c)

$x_s \in \Delta(\mathcal{A}(s))$ \hspace{1cm} (8d)

maximized over all $b \in \mathcal{B}$, where $V(\pi^*, s')$ is the real value computed from the solution of the problem for $s' \in \mathcal{S}$. Then, we define $\pi^*(s) = x_s$. Assumption 2 allows us to use $\geq$ instead of $>$ in constraint (8c) so that the feasible region of the optimization problem is closed. The optimization problem for $s \in \mathcal{S}_i$, i.e., states in layer $h$, relies on the solution $V(\pi^*, s')$ to the optimization problem for states in the next layer $\mathcal{S}_{h+1}$. Therefore, we can efficiently solve (8) for each layer of the state space through backward induction, reducing (8) to an LP.

The best policy in hindsight is the best policy had the follower’s utility function been known from the beginning. Therefore, the solutions to the set of LPs in (8) give the best policy in hindsight and represent the subgame perfect equilibrium of the DSG.

### III. Algorithm

First, we consider the simpler case of learning pure strategies.

In this case, Problem 1 has a simple solution. The leader only needs to learn the function $\varphi : \mathcal{A} \rightarrow \mathcal{B}$ such that

$$
\varphi(a_t) = \arg\max_{b_t} u(a_t, b_t).
$$

Even if the utility function $u$ is unknown, $\varphi$ can be exactly determined by testing each input in $\mathcal{A}$ only once. This strategy does not work for the general case since the continuous space of mixed strategies cannot be enumerated.

Recall that the follower’s utility function follows the relationship $u(a, b) = (f(a, b), \theta^*)$ for a fixed vector $\theta^* \in \mathbb{R}^p$. Every time a mixed strategy $x \in \Delta(\mathcal{A})$ is played and a response $b \in \mathcal{B}$ from the follower is observed, we gain more information about the nature of the follower’s utility function. In particular, we know that $\forall b' \in \mathcal{B}$

$$
\mathbb{E}_{a \sim x} [u(a, b)] \geq \mathbb{E}_{a \sim x} [u(a, b')] \hspace{1cm} (10a)
$$

$$
\mathbb{E}_{a \sim x} [(f(a, b), \theta^*)] \geq \mathbb{E}_{a \sim x} [(f(a, b'), \theta^*)] \hspace{1cm} (10b)
$$

$$
x_T^T (\mathbf{M}_b - \mathbf{M}_{b'}) \theta^* \geq 0 \hspace{1cm} (10d)
$$

If $\theta^*$ is unknown, then (10) lets us interpret $(x, b)$ as a sample for the half-space parameterized by $\theta^*$. Let $\{\{(x_i, b_i)\}_{i \in I} \}$ represent the set of mixed strategies and actions chosen in response from the previous steps of the game. We maintain a *version space* $\Theta \subset \mathbb{R}^p$ of the possible values of $\theta^*$ given the set of previous plays $\{\{(x_i, b_i)\}_{i \in I}$. Specifically, the version space is the convex region

$$
\Theta = \left\{ \theta \in \mathbb{R}^p \mid ||\theta|| = 1 \land \forall i, b' x_T^T (\mathbf{M}_{b_i} - \mathbf{M}_{b'}) \theta \geq 0 \right\}. 
$$

(11)
Before each episode of the game, we solve for an optimistic policy \( \pi_t(s) = x_s \) in the following way. In each state \( s \in S \), we solve for a triple \((x_s, \theta_s, b_s)\) such that if \( \theta_s \) were the true parameterization of \( u \), then the follower would respond to the mixed strategy \( x_s \) with \( b_s \). We constrain \( \theta_s \in \Theta \) to lie within the current version space. The policy is considered optimistic since \( \theta_s \) is allowed to take on any possible value that is consistent with previous observations. The mixed strategy \( x_s \) is computed using backward induction in the following way.

Let \( \epsilon = 2T^{-\frac{1}{2}} \) (see Section IV-B for the derivation of this value). For each state \( s \), we compute the values of \( x_s \) and \( \theta_s \) in the triple along with an optimistic value \( \tilde{V}_t(s) \) as the solution to the optimization problem

\[
\begin{align*}
\max_{x_s, \theta_s} & \quad \tilde{V}_t(s) \\
\text{s.t.} & \quad \theta_s \in \Theta \\
& \quad x_s \in \Delta(A(s)) \\
& \quad \tilde{V}_t(s) = \mathbb{E}_{\sim \sim \sim} \left[ r(s, a, b) + \sum_{s'} P(s, a, b, s') \tilde{V}_t(s') \right] \\
& \quad x_s^T (M_b - M_b') \theta_s \geq \epsilon \quad \forall b' \in B, b' \neq b
\end{align*}
\]  

(12)

maximized over \( b \in B \). That is, we solve (12) for each \( b \in B \) and choose the solution for which \( \tilde{V}_t(s) \) is maximized. The maximizing action becomes the value of \( b_s \) in the triple. We say that the chosen policy is \( \epsilon \)-conservative since constraint (12) enforces an \( \epsilon \)-size margin in the follower’s decision boundary. The \( \epsilon \)-size margin is essential in guaranteeing that we learn a sufficient amount about the value of \( \theta_s \) by further shrinking the version space \( \Theta \) (see Lemma 1 for more details).

Notice that the optimization problem for \( s \in S_h \), i.e., states in layer \( h \), relies on the solution \( \tilde{V}_t(s') \) to the optimization problem for states in the next layer \( S_{h+1} \). Therefore, we can efficiently solve (12) for each layer of the state space by computing the final layer \( S_1 \) first and then iterating backward.

Function \textproc{GetPolicy} in Algorithm 1 shows how the policy is computed. These solutions give rise to a policy \( \pi_t(s) = x_s \) and an associated optimistic value \( \tilde{V}_t(s) \) at each state.

After computing a policy \( \pi_t \) for episode \( t \) in the game, it is used to play mixed strategies for the course of the episode. Beginning with the initial state \( s_1 \), the leader plays mixed strategy \( \pi_t(s_1) \). After observing the follower’s response \( b \) to \( \pi_t(s_1) \), the algorithm updates the version space \( \Theta \) with the new information. Regardless of which action the policy \( \pi_t \) expected the follower to respond with, it is now known that \( x(\theta_b - \theta_b') \geq 0 \). Therefore, the version space can be updated to

\[
\Theta \cap \left\{ \theta \in \mathbb{R}^p \mid \| \theta \| = 1 \land \forall b' \quad x^T (M_b - M_b') \theta \geq 0 \right\}.
\]  

(13)

The function \textproc{Update} in Algorithm 1 encapsulates this update rule. Fig. 2 outlines how the various components of the algorithm are used to compute and update policies. Fig. 3 gives a pictorial representation of the learning scheme and DSG.

**Remark 1:** The procedure \textproc{GetPolicy} requires solving \(|S| \cdot |B| \) copies of the nonconvex quadratic program in (12) to compute the triple \((x_s, \theta_s, b_s)\) for each state in \( S \).

**IV. REGRET ANALYSIS**

In this section, we establish a high-probability regret bound for the proposed scheme outlined in Fig. 2.

**Theorem 1:** Fix confidence parameter \( \delta > 0 \). The regret of the learning scheme in Fig. 2 satisfies

\[
R_T \leq (T)^{1 - \frac{1}{2}} \left( d\sqrt{mn(1 + \sqrt{nH})H + H} \right) + H \sqrt{\frac{T}{2} \ln \left( \frac{1}{\delta} \right)}
\]  

(14)

with probability at least \( 1 - \delta \) for some constant \( d \in \mathbb{R} \).

**Remark 2:** Most notably, the sublinearity in \( T \) of the regret depends on the dimension \( p \) of the follower’s parameter space.

---

**Algorithm 1:** Subroutines Used in the Learning Scheme. \( P_c \) Denotes the Program in (12).

1: function \textproc{GetPolicy}(t, \Theta)
2: for \( s \in S \) do
3: \( \pi_t(s) \leftarrow 0 \)
4: \( \tilde{V}_t(s) \leftarrow 0 \)
5: end for
6: for \( b \in H \) to 1 do
7: for \( s \in S_h \) do
8: \( \pi_t(s), \tilde{V}_t(s) \leftarrow P_c(\Theta, \tilde{V}_t(s)) \)
9: end for
10: end for
11: return \( \pi_t \)
12: end Function
13: function \textproc{Update}(x, b, \Theta)
14: return \( \Theta \cap \left\{ \theta \in \mathbb{R}^p \mid \| \theta \| = 1 \land \forall b' \quad x^T (M_b - M_b') \theta \geq 0 \right\} \)
15: end Function

---

**Parameters:** Discrete-time dynamic Stackelberg game \( \langle S, A, B, r, u, P \rangle \) and time horizon \( T \).

\( \Theta \leftarrow \{ \theta \in \mathbb{R}^p \mid \| \theta \| = 1 \} \).

For all episodes \( t = 1, 2, \ldots, T \), repeat

\( \pi_t \leftarrow \textproc{GetPolicy}(t, \Theta) \).

For all steps \( h = 1, 2, \ldots, H \), repeat

1) The learner observes the current state \( s \in S_h \).
2) The learner selects mixed strategy \( x \leftarrow \pi_t(s) \).
3) The leader observes the follower’s action \( b \) and updates \( \Theta \leftarrow \textproc{Update}(x, b, \Theta) \).
4) An action \( a \sim x \) is sampled.
5) The learner obtains reward \( r(s, a, b) \) and observes \( s' \sim P_s(\cdot \mid s, a, b) \).

---

**Fig. 2:** Procedure for using the learning scheme to compute policies in a discrete-time DSG.

---
However, the regret depends only polynomially in the remaining parameters of the game and has a complete lack of dependence on the size of the state space. In Section V, we experimentally show the dependence of the regret on $p$ and the independence on the size of the state space.

Remark 3: We further note that the result of Theorem 1 holds with high probability. Typically, high-probability regret bounds are more challenging to derive than expected regret bounds in online learning [25], [26], [27]. As we will see, we establish the result of Theorem 1 by a careful application of Azuma’s inequality of Martingales [28]. Given this result, an upper bound on the expected regret of the proposed scheme can be obtained by straightforward integration of the tail of the high-probability regret bound that we establish in Theorem 1 (see, e.g., [29]).

Next, we give a high-level idea of our proof techniques, which is then followed by the formal proof of Theorem 1.

A. Proof Outline

At the beginning of each episode of the game, we optimistically solve for the optimal $\epsilon$-conservative policy $\pi_t$ by calling GETPOLICY. Each mixed strategy $x_s = \pi_t(s)$ is associated with a choice $\theta_s \in \Theta$ and expected response $b_s \in B$. Let $b_s$ be the actual action that the follower would take in response to mixed strategy $x_s$. If $b_s^* = b_s$, $\forall s \in S$, then we will show that policy $\pi_t$ is at least as good as the optimal $\epsilon$-conservative mixed since $\theta^* \in \Theta$ and, thus, one of the possible values of $\theta$ consistent with previous observations.

If instead $b_s \neq b_s^*$ for some $s \in S$, then we say that $\pi_t$ has some probability of making a mistake. Such a scenario is called a mistake since the computed policy $\pi_t$ with associated triple $(x_s, \theta_s, b_s)$ expects the follower to play $b_s$ in response to mixed strategy $x_s$. If $b_s$ is not actually played in response to $x_s$, then the computed value $V_t(s)$ is overly optimistic. In Lemma 3, we show that playing policy $\pi_t$ induces regret proportional to the probability of making a mistake while executing $\pi_t$.

The resulting observation $(x_s, b')$ after making a mistake shrinks the version space $\Theta$ by inducing new hyper-plane constraints. In Lemma 1, we show that, since $x_s$ is chosen to be $\epsilon$-conservative, whenever it leads to a mistake, $\Theta$ is shrunk by at least some fixed volume. In Lemma 2, we show that since the initial volume of $\Theta$ is finite, the total number of mistakes is bounded.

The learning scheme induces two sources of regret: 1) regret from choosing an $\epsilon$-conservative policy for each episode and 2) regret from the probability of making a bounded number of mistakes. In Lemma 5, we upper bound the regret resulting from choosing an $\epsilon$-conservative policy. Using the fixed upper bound on the number of mistakes induced by policy $\pi_t$, we derive a probabilistic upper bound on the cumulative probability that policy $\pi_t$ leads to a mistake using Azuma’s inequality of Martingales. Finally, at the end of Section IV, we use this sequence of lemmas to prove Theorem 1.

B. Proof of Theorem 1

In this section, we establish the proof of Theorem 1 by providing a number of intermediate lemmas, which we discussed in Section IV-A. Proofs of the intermediate lemmas are left to the Appendix.

Before showing that the learning scheme makes a bounded number of mistakes in Lemmas 1 and 2, we show that the follower’s reward function can be scaled without affecting its policy. This allows us to assume, without loss of generality, that the difference of feature matrices $| | M_b - M_{b'} | |_{op} \leq 1$ for any $b, b' \in B$.

Proposition 1: The follower’s policy is invariant under feature mappings up to scalar multiplication. That is, if $\varphi(x)$ and $\varphi'(x)$ denote the policies for feature mappings $f(a, b)$ and $f'(a, b) = c \cdot f(a, b)$, respectively, for $c \in \mathbb{R}$, then $\varphi(x) = \varphi'(x)$.

Given Proposition 1, the feature map $f$ can be scaled arbitrarily, without loss of generality.

We proceed with the first two lemmas by first showing that samples gathered when the learning schemes makes mistakes are guaranteed to shrink the version space by a fixed amount. We use this to show that policies resulting from the optimization
problem in (12) throughout the execution of the learning scheme only make a bounded number of mistakes.

**Lemma 1:** Let \((x_s, \theta_s, b_s)\) for any \(s \in S\) be the solution from solving (12). If the solution \((x_s, \theta_s, b_s)\) makes a mistake during execution in the sense that action \(b_s\) is predicted by \(\theta_s\) but \(b'\) is actually played in response to \(x\), then \(\text{UPDATE}(x, b', \Theta)\) shrinks \(\Theta\) by at least \(\Theta \cap B_s(\theta_s)\).

Since the volume of the initial version space \(\Theta\) is finite, Lemma 1 bounds the total number of possible mistakes.

**Lemma 2:** Solutions \((x_s, \theta_s, b_s)\) found by the learning scheme produce mistakes during execution at most \((\frac{2}{\epsilon})^{p-1}\) times.

Now that we have given an upper bound on the number of mistakes that mixed strategies resulting from the learning scheme can make, we proceed by bounding the regret induced by solving for an \(\epsilon\)-conservative policy.

Define the **optimal \(\epsilon\)-conservative policy** \(\pi^\epsilon : S \to \Delta(A)\) in each state \(s \in S\) as the solution to the following optimization problem:

\[
\max_{x_s} V(\pi^\epsilon, s) \tag{15a}
\]

s.t. \(V(\pi^\epsilon, s) = \mathbb{E}_{a \sim x_s} \left[ r(s, a, b) + \sum_{s'} P(s, a, b, s')V(\pi^\epsilon, s') \right] \tag{15b}\)

\(x_s^T (M_b - M_{b'}) \beta^* > \epsilon \quad \forall b' \in B \tag{15c}\)

\(x_s \in \Delta(A(s)) \tag{15d}\)

maximized over all \(b \in B\). Then, \(\pi^\epsilon(s) := x_s\). Notice that the solutions to (15) can be computed using backward induction in the same way as in (12).

**Definition 2:** Let \(V(\pi, s)\) represent the **actual expected value** obtained during an episode from playing policy \(\pi\) beginning in state \(s\).

We say that a trace \(\tau\) resulting from a policy \(\pi_t\) makes a mistake if \(\tau\) results in playing a policy \(x_s\) at some state \(s \in S\) with the expectation that action \(b_s\) will be played (i.e., \(x_s^T (M_{b_s} - M_{b_s'}) \theta \geq 0\) for all \(b' \in B\)) but in reality, a different action is played (i.e., \(x_s^T (M_{b_s} - M_{b_s'}) \theta^* \leq 0\) for some \(b' \in B\)). First off, notice that if no traces obtained by following policy \(\pi_t\) from state \(s\) make a mistake, then \(V(\pi_t, s) = V_t(s)\). Otherwise, we have the following lemma.

**Lemma 3:** Let \(\lambda_t(s)\) be the probability that \(\pi_t\) makes a mistake during the episode beginning from state \(s\). Then

\[
V(\pi^\epsilon, s) > V(\pi_t, s) \geq V(\pi^\epsilon, s) - H \cdot \lambda_t(s) \tag{16}
\]

The difference in optimality between \(V(\pi^*, s)\) and \(V(\pi^\epsilon, s)\) differs only by the difference in feasibility region induced by relaxing (15c) to \(x_s^T (M_{b_s} - M_{b_s'}) \theta \geq 0\). Therefore, the sub-optimality of \(V(\pi_t, s)\) is bounded by the maximum difference in value induced by two solutions that differ by the Hausdorff distance between the feasibility regions. The Hausdorff distance is realized at the critical points of the feasibility regions. Let \(C \subset B, |C| = q\) represent a critical boundary of the feasibility space induced by the relaxation of (15c). That is, there exists some \(x^*\) such that \(x^T (M_b - M_{b'}) \theta^* = 0\) for all \(b' \in C\).
making mistakes. By Lemma 5, we continue to get
\[ R_T \leq T \left( \epsilon d \sqrt{mn(1 + \sqrt{nH})} + H \sum_{t=1}^{T} \lambda_t(s_1) \right). \] (22)

The final step is to use the fixed upper bound on the number of mistakes to give a probabilistic upper bound on the cumulative probability of making a mistake.

Let \( X_t \) be the indicator random variable for the event that \( \pi_t \) makes a mistake during execution on step \( t \) of the game. Let \( \lambda_i = \lambda_i(s_1) \). By construction, \( E[X_t] = \lambda_t \). Notice that \( \lambda_t \) depends on the outcomes of \( X_1, X_2, \ldots, X_{t-1} \).

Define the martingale \( M_t = \sum_{i=1}^{t} (X_i - \lambda_i) \) with filtration \( F_t = \sigma(\{X_j, \lambda_j\}_{j=1}^{t}) \). The sequence \( \{M_t\} \) is indeed a martingale since
\[ E[M_{t+1} \mid F_t] = E[X_{t+1} - \lambda_{t+1} \mid F_t] + M_t = M_t \] (23)

where the final equality comes from the fact that, by construction, the expectation of \( X_{t+1} \) is \( \lambda_{t+1} \) after \( X_1, \ldots, X_t \) have been observed. Since \( |M_{t+1} - M_t| = |X_{t+1} - \lambda_{t+1}| \leq 1 \), by Azuma’s inequality [28], we have
\[ \Pr[|M_t| \geq \alpha] \leq \exp \left(-\frac{\alpha^2}{2T}\right). \] (24)

From Lemma 2, the actual number of mistakes \( \sum_{i=1}^{T} X_i \) is upper bounded by \( \left( \frac{2}{\epsilon} \right)^{p-1} \). Therefore
\[ M_T = \sum_{i=1}^{T} X_i - \sum_{i=1}^{T} \lambda_i \leq \left( \frac{2}{\epsilon} \right)^{p-1} - \sum_{i=1}^{T} \lambda_i. \] (25)

Combining this with (24), we obtain
\[ \Pr \left[ \sum_{i=1}^{T} \lambda_i \geq \left( \frac{2}{\epsilon} \right)^{p-1} + \alpha \right] \leq \exp \left(-\frac{\alpha^2}{2T}\right). \] (26)

Returning to the regret, this gives
\[ R_T \leq T \left( \epsilon d \sqrt{mn(1 + \sqrt{nH})} + H \left( \frac{2}{\epsilon} \right)^{p-1} + \alpha \right) \] (27)

with probability at least \( 1 - e^{-\frac{\alpha^2}{2T}} \). Choosing \( \epsilon = 2T^{-\frac{1}{p}} \) gives
\[ R_T \leq 2d \sqrt{mn(1 + \sqrt{nH})} H (T)^{1 - \frac{1}{p}} + H(T)^{1 - \frac{1}{p}} + \alpha H \] (28a)

with probability at least \( 1 - e^{-\frac{\alpha^2}{2T}} \). Taking \( \alpha = \beta \sqrt{T} \) gives
\[ R_T \leq (T)^{1 - \frac{1}{p}} \left( 2d \sqrt{mn(1 + \sqrt{nH})} H + H \right) + \beta \sqrt{T} H \] (28b)

with probability at least \( 1 - e^{-\frac{\alpha^2}{2T}} \). For any choice of \( \beta \), letting \( \delta = e^{-\frac{\alpha^2}{2T}} \), we can get the alternate representation
\[ R_T \leq (T)^{1 - \frac{1}{p}} \left( 2d \sqrt{mn(1 + \sqrt{nH})} H + H \right) + H \sqrt{2T \ln \left( \frac{1}{\delta} \right)} \] (29)

with probability at least \( 1 - \delta \).

### Algorithm 2: An Any-Time Version of the Learning Scheme.

**Require:** Dynamic Stackelberg game \((\mathcal{S}, \mathcal{A}, \mathcal{B}, r, u, P)\).

1: Initialize \( T_0 \).
2: for \( i \in \{0, \ldots, n\} \) do
3: \( T_i \leftarrow 2T_0 \)
4: \( \text{LEARN}(T_i) \)
5: end for

### C. Anytime Sublinear Regret for DSGs

While we showed in the previous section that the learning scheme outlined in Fig. 2 enjoys a high-probability sublinear regret, the algorithm relies on knowing the number of episodes \( T \). In what follows, using the doubling trick [30], we show how to adapt the learning scheme into an any-time algorithm that does not require knowing \( T \) in advance. That is, even if the horizon \( T \) is unknown, the adapted version of our learning scheme will achieve the same regret bound as the standard learning scheme, with probability \( 1 - \delta \), for any confidence parameter \( \delta > 0 \).

The adapted learning scheme, outlined in Algorithm 2, operates by learning over increasingly large time segments \( \{T_i\} \), starting from an initial segment \( T_0 \). To make our analysis simpler, we first consider the scheme in which progress made in a segment \( T_i \) is discarded once segment \( T_{i+1} \) begins.

Let \( T_i = 2^i \) and \( T_{\text{true}} = \sum_{i=0}^{n} T_i \). \( CR(t) \) is the cumulative regret of Algorithm 2 at time \( t \) and \( R(t) \) is the regret of the learning scheme outline in Fig. 2 (cf. Theorem 1). Then, it holds with probability at least \((1 - \delta)^n\) that

\[ CR(T_{\text{true}}) = \sum_{i=0}^{n} R_{T_i} \] (31a)

\[ \leq \sum_{i=0}^{n} (T_i)^{1 - \frac{1}{p}} \left( d \sqrt{mn(1 + \sqrt{nH})} H + e^{p-1} H \right) + H \sqrt{T_i \ln \left( \frac{1}{\delta} \right)} \] (31b)

\[ = \left( d \sqrt{mn(1 + \sqrt{nH})} H + e^{p-1} H \right) \sum_{i=0}^{n} (T_i)^{1 - \frac{1}{p}} \] (31c)

\[ + H \sqrt{\frac{1}{2} \ln \left( \frac{1}{\delta} \right)} \sum_{i=0}^{n} \sqrt{T_i}. \] (31d)

Then, using a simple geometric sum formula, one can establish that \( CR(T_{\text{true}}) = \tilde{O}(T_{\text{true}}^{1 - \frac{1}{p}}) \), with probability at least \((1 - \delta)^n \geq 1 - n\delta \). Therefore, by rescaling \( \delta \) to \( \delta/n \), we establish the intended result.

Now, we consider the case in which progress in the learning scheme is carried over between time segments. In this case, instead of discarding the half-spaces accumulated during previous segments, we continue to use those half-spaces to restrict the possible value of \( \theta^* \). The cumulative regret incurred during segment \( T_i \) is then upper bounded by \( R_{T_i} \), since the volume of the version space \( \Theta \) when starting segment \( T_i \) is at most the volume of the initial version space \( \Theta_0 \). In fact, it is likely smaller, since
mistakes made in previous segments would have incurred new half-spaces that shrink the size of $\Theta$ even before segment $T_i$ begins.

V. EXPERIMENTAL RESULTS

In this section, we present experimental results on the regret that our algorithm incurs over discrete-time DSGs with varying parameters. First, we discuss results related to the poaching example from the introduction, and then, we give results averaged over randomly generated DSG instances to show performance as parameters of the DSG vary. Our implementation uses Gurobi [31] to compute solutions to the quadratic program in (12).

To demonstrate the performance of our algorithm, we report the average regret over time. At time step $t$, the average regret is calculated as

$$\frac{1}{tH} \sum_{i=1}^{t} \sum_{h=1}^{H} R_{i,h}^\pi - R_{i,h}^\pi_\ast$$

where $R_{i,h}^\pi$ and $R_{i,h}^\pi_\ast$ is the reward that the optimal policy and learning policy, respectively, receive at step $h$ of episode $i$. To compare the performance of multiple different policies, we also report the average cumulative reward over time. At time step $t$, the average cumulative reward is calculated as

$$\frac{1}{tH} \sum_{i=1}^{t} \sum_{h=1}^{H} R_{i,h}^\pi$$

where $R_{i,h}^\pi$ is the reward that policy $\pi_i$ receives at step $h$ of episode $i$.

To the best of the authors’ knowledge, no other algorithms exist for directly solving Problem 1. However, by disregarding the reward and transition structure induced by the Stackelberg game, a discrete-time DSG can be reduced to an MDP with an unknown reward and transition functions in the following way.

Let $(S, A, B, r, u, P)$ be a discrete-time DSG. Consider the auxiliary reward function $R : \Delta(A) \times S \to \mathbb{R}$ defined in (3) induced by the follower’s policy. An auxiliary transition function $P' : S \times \Delta(A) \times S$ can be defined in a similar way as

$$P'(s, x, s') = \sum_{a \in \mathcal{A}} [P(s, a, \varphi(x), s')] .$$

Therefore, the game can be reduced to the MDP with parameters $(S, \Delta(A), R, P')$. Notice that the action space is the continuous space of mixed policies and that if the follower’s utility function is unknown, then $R$ and $P'$ are also unknown.

By disregarding the reward and transition structure given by $R$ and $P'$, respectively, learning in this setting can be done with reinforcement learning. Q-learning is a popular model-free reinforcement learning algorithm suitable for this scenario. Since the action space of the resulting MDP is continuous, we discretize it and use a tabular implementation of Q-learning.

A. Poaching Example

Recall the motivating example from Section I: park rangers are responsible for protecting a geographical area containing different types of animals from illegal poachers. The geographical area is split into distinct regions, each of which contains a different density of the animals. Each month (i.e., the state), the park rangers choose a mixed policy to decide which of the regions to patrol. This mixed policy can be implemented through a randomization device that picks which specific region to patrol on each day according to the probability distribution specified by the mixed strategy. We assume the worst case that after observing the ranger’s policy for the first few days of the month, the poachers are able to accurately estimate the distribution of the ranger’s random patrols and lay snares in any one of the regions in response.

The park rangers may not know exactly which types of animals the poachers are attempting to poach. Therefore, we model the poachers’ reward function as an unknown linear combination of the probability of poaching each type of animal with a penalty if the poacher is caught. Let $D : [N] \to [0, 1]^M$ represent a density function such that $[D(n)]_m$ represents the density of animal $m$ in subregion $n$. The poacher’s utility function can, therefore, be described as

$$u(a, b) = \langle f(a, b), \theta' \rangle$$

with unknown weight vector $\theta'$ and known feature function

$$f(a, b) = \begin{cases} (0, \ldots, 0, -1), \text{ if } a = b \\ (D(b), 0), \text{ otherwise} \end{cases} .$$

The last index in $f(a, b)$ describes whether or not the poacher is caught by the part rangers. Therefore, the last index in $\theta'$ describes the severity of the ranger getting caught and the first $M$ indices describe the payoff for successfully poaching different types of animals.

The park ranger’s reward function correlates directly with the fluctuating severity of the various animals being poached throughout the year. Let $C : [12] \to [0, 1]^M$ represent the severity of poaching each of the $M$ types of animals per month. Let $s = (s_1, s_2)$ represent the park ranger’s state, where $s_1$ is the current month and $s_2$ is the remaining budget. Then, the park rangers’ reward function is

$$r(s, a, b) = \begin{cases} c, \text{ if } a = b \\ -C(s_1) \cdot D(b), \text{ otherwise} \end{cases} .$$

where $c$ is a constant describing the value of catching and arresting a poacher. The set of actions $A(s)$ available to the park rangers from state $s$ are the regions that keep the park rangers under budget. Notably, the poacher’s utility is independent of the state; they do not care about how the severity of poaching changes throughout the month.

We explore the performance of our learning scheme on an instance of the poaching example with a budget of six, a horizon of four, and three different types of animals (giving $p = 4$). The park is split into four different regions, two of which cost one unit of the budget to patrol, and two that cost two units of budget to patrol. The poaching density for each type of animal is randomly chosen in each region, and the poaching severity for each type of animal is randomly chosen for each region. The poacher’s preference $\theta'$ is randomly chosen and unknown to the leader.
LAUFFER et al.: NO-REGRET LEARNING IN DYNAMIC STACKELBERG GAMES

Fig. 4. Average cumulative reward over time of the learning scheme, a random policy, and the best policy in hindsight on an instance of the poaching example. The best policy in hindsight is the optimal policy had the follower’s reward function been known ahead of time. Regret is given by taking the difference between the best policy in hindsight.

Fig. 4 shows a comparison of the performance of our learning scheme against the best policy in hindsight and a discretized implementation of Q-learning over a time horizon of 30. Since policies in the poaching example are stochastic, the results in Fig. 4 show the mean over ten samples with the shaded interval representing one standard deviation of the mean.

B. Randomly Generated Instances

Each of the randomly generated DSG instances has four actions available for both the leader and the follower ($n = 4$ and $m = 4$). In order to report the average performance in different scenarios, numerical values in the game are generated randomly, and results are averaged over ten different games. In particular, rewards and transitions between layers in the state space are generated uniformly at random. The state spaces resemble a tree, with an increasing number of states in each layer.

Fig. 5 shows the average regret that our algorithm incurs over a horizon of $T = 20$ with varying dimensions $p$ of the follower parameter space. As outlined above, the numerical values are generated randomly and the state space has $(1, 2, 4, 8, 16)$ states in each layer. In agreement with the theoretical analysis, the asymptotic behavior of the learning agent approaches the optimal policy more slowly with a larger value of $p$.

Fig. 6 shows the average regret that our algorithm incurs over a horizon of $T = 20$ with varying size state spaces but fixed $p = 4$. The number of states in each layer is varied according to the chart in Table I. In agreement with the theoretical analysis, the plot indicates that the regret of our algorithm is independent of the size of the state space.

For our final experimental result, we also compare against a Monte Carlo tree search (MCTS)-based algorithm for learning in repeated Stackelberg games with unknown opponents [10]. Because the MCTS algorithm only works for stateless Stackelberg games, we need to run a separate instance of the MCTS algorithm for each state in the game. This approach has two immediate drawbacks compared to our algorithm: 1) it is unable to plan ahead over multiple time steps and 2) it is not able to share experience for learning across different states in the game. Our implementation of the MCTS algorithm uses a discretization of $K = 4$ and 2000 Monte Carlo samples per state.

Fig. 7 compares the average cumulative reward between the best policy in hindsight, our algorithm, Q-learning, the

---

**TABLE I**

| Number of States in Each Layer of a DSG Instance Having Four Actions Available for Both the Leader and the Follower ($n = 4$ and $m = 4$) |
|---|---|---|---|---|
| $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
| 1 | 2 | 2 | 2 | 2 |
| 1 | 2 | 4 | 4 | 4 |
| 1 | 2 | 4 | 8 | 8 |
| 1 | 2 | 4 | 8 | 16 |
MCTS algorithm, and a policy that randomly chooses mixed policies. The Q-learning implementation discretizes the action space into mixed strategies of the form $(z_1, z_2, z_3, z_4)$ for integers $z_1, z_2, z_3, z_4 \in \mathbb{Z}$ such that $\sum_i z_i = 10$. The state space is partitioned into layers of size $(1,2,2,2,2)$.

As demonstrated in Fig. 7, Q-learning and the MCTS algorithm significantly underperform in comparison to our algorithm. Larger state spaces lead to even greater discrepancies in performance since the amount of time it takes Q-learning and the MCTS algorithm to learn scales with the size of the state space, whereas our algorithm provably does not.

We also experimentally evaluate our learning algorithm against a setting in which the value of $\theta^*$ can be estimated a priori with varying amounts of Gaussian error. Fig. 8 shows the performance of the optimal policy in hindsight, our learning algorithm, and policies that compute the optimal policy given an unbiased noisy estimate of $\theta^*$, where the standard deviation of the Gaussian error ranges from $10^{-4}$ to $10^0$. As the theory suggests, using a fixed estimate of $\theta^*$ with any error incurs linear regret in expectation, whereas our learning algorithm incurs sublinear regret. Empirically, the performance of our learning algorithm quickly outpaces any fixed estimate, even if the estimate contains little noise.

VI. CONCLUSION

We introduce a new class of game, called a discrete-time DSG, by unifying the sequential decision-making aspects of an MDP and the asymmetric interaction between two strategic agents from a Stackelberg game. DSGs extend standard repeated Stackelberg games to scenarios with dynamic reward structures and action spaces. We study DSGs in an online learning setting and give a novel no-regret learning algorithm for playing in a DSG against a follower with unknown utility function. Experimental results show that even in practice, our algorithm outperforms model-free reinforcement learning approaches for solving DSGs. Moreover, our algorithm achieves regret independent of the size of the state space, providing a scalable solution in environments with large state spaces.

A. Future Work

1) Tightness of the Analysis: The most limiting aspect of our result seems to be the nature of the regret’s dependence on the dimension of $\theta^*$, the parameter of the follower’s utility function. Lemma 2 achieves an upper bound on the number of mistakes that is linear in the volume of the initial version space of $\theta$. A closely related problem to this subroutine of the main algorithm is the problem of exactly learning an unknown half-space through query synthesis, for which there exist algorithms that achieve sample complexity that is logarithmic in the volume of the initial version space [32]. Bridging the gap between the sample efficiency of our subroutine and the efficiency of this closely related problem could be a fruitful method for improving the regret bound of our result.

2) Function Class of the Follower’s Utility: Our algorithm relies on the fact that the follower’s utility function is linearly parameterized. Without this assumption, estimating the follower’s utility function is challenging. For any prior observation $(x, b)$, we know that for all $b'$ in $B$,

$$\mathbb{E}_{a \sim x}[u(a, b)] \geq \mathbb{E}_{a \sim x}[u(a, b')]$$

(38a)

$$\langle x, \theta_b \rangle \geq \langle x, \theta_{b'} \rangle$$

(38b)

for some vectors $\theta_b$. Estimating the decision boundaries induced by the vectors $\theta_b$ from queries requires exactly learning this multiclass classification problem. Solving the posed learning problem for discrete-time DSGs with a generalized follower utility function would likely require incorporating some form of online learning algorithm for the multiclass classification problem.
3) Computational Complexity: In order to compute a policy for each episode, our algorithm requires solving $|S| \cdot |B|$ copies of the nonconvex quadratic program in (12). In general, solving a nonconvex quadratic program is NP-hard, which could pose limitations when scaling to larger problem instances. Designing more efficient algorithms for Problem 1 could be a useful direction for future research.

APPENDIX A

PROOF OF PROPOSITION 1

Let $c \in \mathbb{R}$ and define an augmented feature function $f'(a, b) = c \cdot f(a, b)$. A corresponding augmented utility function for the follower can be defined as

$$u'(a, b) = \langle f'(a, b), \theta^* \rangle = c \cdot u(a, b). \tag{39}$$

The follower’s corresponding augmented policy is therefore

$$\varphi'(x) = \arg \max_{b \in B} \mathbb{E}_{a \sim x} [c \cdot u(a, b)].$$

Identical to the original utility function.

APPENDIX B

PROOF OF LEMMA 1

From (12), we see that $x_T^+(B_{b^*} - B_{b^*}) \theta_s \geq \epsilon$. Since $b^*$ was the optimal choice for the follower, we have $x_T^+(B_{b^*} - B_{b^*}) \theta_s^* \leq 0$. Combining these, we obtain

$$x_T^+(B_{b^*} - B_{b^*}) \theta_s - x_T^+(B_{b^*} - B_{b^*}) \theta^* \geq \epsilon$$

$$\|x_T^+(B_{b^*} - B_{b^*})\| \cdot \|\theta_s - \theta^*\| \geq \epsilon$$

$$\|x_T^+\| \cdot \|\theta_s - \theta^*\| \geq \epsilon$$

$$\|\theta_s - \theta^*\| \geq \epsilon$$

where step (a) is by the Cauchy–Schwarz inequality, step (b) follows from Proposition 1, and step (c) follows from the fact that $x_T^+$ is a probability vector. Therefore, $\theta^* \notin B_s(\theta_s)$, and the proof is complete.

APPENDIX C

PROOF OF LEMMA 2

Since we shrink $\Theta$ by at least $\Theta \cap B_s(\theta_s)$ with mistake $(x_s, \theta_s, b_s)$, we know that any future mistakes cannot fall within the ball $B_s(\theta_s)$. Let $\theta_1$ and $\theta_2$ be two estimates for the half-space $\theta^*$ that induced a mistake during a previous observation. It must be that $B_{\frac{1}{2}}(\theta_1) \cap B_{\frac{1}{2}}(\theta_2) = \emptyset$. Therefore, the number of mistakes is upper bounded by the number of $\frac{1}{2}$-balls that can fit in the initial space $\theta_0 = \{\theta \in \mathbb{R}^p \mid \|\theta\| = 1\} = \{\theta \in \mathbb{R}^{p-1} \mid \|\theta\| \leq 1\}$. Let $V_{p-1}(r)$ denote the $(p-1)$-dimensional volume of a $(p-1)$-sphere with radius $r$. An upper bound on the number of mistakes is, therefore, given by the ratio between the volumes

$$\frac{V_{p-1}(1)}{V_{p-1}(\frac{1}{2})} = \frac{\alpha}{\left(\frac{\alpha}{2}\right)^{p-1}} = \left(\frac{2}{\epsilon}\right)^{p-1} \tag{41}$$

where $\alpha = \frac{\pi(\frac{1}{2})}{\Gamma(\frac{p}{2}+1)}$ and $\Gamma$ is the gamma function.

APPENDIX D

PROOF OF LEMMA 3

The inequality $V(\pi^*, s) \geq V(\pi_t, s)$ is obvious since $\pi^*$ is the optimal policy. Let $\pi \sim \pi_t(s)$ represent a trace, i.e., a sequence of state–action pairs sampled from following policy $\pi_t$ beginning in state $s$. Let $V(\pi)$ represent the actual reward obtained from $\pi$ and let $\tilde{V}_t(\pi)$ represent the estimated reward obtained from $\pi$, i.e., the value that $\tilde{V}_t(s)$ expects outcome $\tau$ to achieve. Notice that $\tilde{V}_t(\pi) = V(\pi)$ if $\pi$ does not make a mistake. Then, $V(\pi_t, s) = \mathbb{E}_{\tau \sim \pi_t(s)}[V(\pi)]$. Let $\mu(\pi)$ be a Boolean value that is true if and only if $\pi$ does not make a mistake under the learning policy. Then, we have

$$V(\pi_t, s) = \mathbb{E}_{\tau \sim \pi_t(s)}[V(\pi) \mid \mu(\pi)] \cdot \mathbb{P}[\mu(\pi)]$$

$$+ \mathbb{E}_{\tau \sim \pi_t(s)}[V(\pi) \mid \neg \mu(\pi)] \cdot \mathbb{P}[\neg \mu(\pi)]$$

$$= \mathbb{E}_{\tau \sim \pi_t(s)}[V(\pi) \mid \mu(\pi)] (1 - \lambda_t(s))$$

$$+ \mathbb{E}_{\tau \sim \pi_t(s)}[V(\pi) \mid \neg \mu(\pi)] \lambda_t(s)$$

$$\geq \mathbb{E}_{\tau \sim \pi_t(s)}[V(\pi) \mid \mu(\pi)] (1 - \lambda_t(s))$$

$$\geq \mathbb{E}_{\tau \sim \pi_t(s)}[\tilde{V}_t(\pi) \mid \mu(\pi)] (1 - \lambda_t(s))$$

$$\geq \mathbb{E}_{\tau \sim \pi_t(s)}[\tilde{V}_t(\pi) - H \cdot \lambda_t(s)]$$

$$\geq \tilde{V}_t(s) - H \cdot \lambda_t(s)$$

Inequality (a) holds because rewards are normalized to be positive, equality (b) comes from the fact that $\tilde{V}_t(\pi) = V(\pi)$ on traces $\tau$ that the learning policy does not make a mistake on, and (c) comes from the fact that $\theta^* \in \Theta$ so $\tilde{V}_t(s) \geq V(\pi^*, s)$.

APPENDIX E

PROOF OF LEMMA 4

Let $A = (v_1 \ldots v_q)$. We just need to show that there exists some $\nu$ such that $x^T \nu = 0$ and $\nu \notin \text{span}\{v_1, \ldots, v_q\}$. We will also show that $\nu$ can be chosen such that the resulting matrix $A' = (A \nu)$ has $\sigma_{\text{min}}(A') = \sigma_{\text{min}}(A)$. Then, through repeated application of this result, the lemma holds.
Let $x^\perp$ denote the complement space of $x$. For the sake of contradiction, suppose that no such vector exists. Then, $x^\perp \subset \operatorname{span}\{v_1, \ldots, v_q\}$. But since $\dim(x^\perp) = n - 1$, it must be that $x \in \ker\{v_1, \ldots, v_q\}$ since the kernel space is non-trivial. However, this is a contradiction, since $x^T M \neq 0$ by definition. Therefore, there exists $v$ such that $x^T v = 0$ and $v \notin \operatorname{span}\{v_1, \ldots, v_q\}$. Let $A' = (A \ v)$. Then
\[(A')^T A' = \begin{pmatrix} A^T & 0 \\ v^T & 0 \end{pmatrix} = \begin{pmatrix} A^T A & 0 \\ 0 & v^T v \end{pmatrix}. \tag{42}\]

Since the singular values of $A'$ are the square roots of the eigenvalues of $(A')^T A'$, we have that $\sigma_{\min}(A') = \min_i(\sigma_{\min}(A), ||v||)$. Since $||v||$ can be arbitrarily chosen, we can construct $A'$ such that $\sigma_{\min}(A') = \sigma_{\min}(A)$ holds.

**Appendix F**

**Proof of Lemma 5**

Define vectors $v^*_s, b \in \mathbb{R}^n$ such that
\[v^*_s, b_a = \sum_{s'} P(s, a, b, s') V(\pi^*, s')\]
and vector $r_{s, b} \in \mathbb{R}^n$ such that
\[r_{s, b}_a = r(s, a, b).\]

We have that $||x^T - x^*|| \leq \sqrt{m} \cdot \epsilon \cdot d$ from (19), which in turn gives
\[V(\pi^*, s) - V(\pi^*, s) = x^T (r_{s, b} + v^*_s, b) - x^* (r_{s, b} + v^*_s, b) = (x^T - x^*) r_{s, b} + x^T v^*_s, b - x^* v^*_s, b = (x^T - x^*) r_{s, b} + (x^T v^*_s, b - x^* v^*_s, b) + x^T v^*_s, b - x^* v^*_s, b = (x^T - x^*) r_{s, b} + x^T (v^*_s, b - v^*_s, b) + (x^T - x^*) v^*_s, b \leq ||x^T - x^*|| ||r_{s, b}|| + ||x^T || ||v^*_s, b - v^*_s, b|| + ||x^T - x^*|| ||v^*_s, b|| \leq \epsilon \cdot d \cdot \sqrt{m} ||r_{s, b}|| + \epsilon \cdot d \cdot \sqrt{m} \cdot \sqrt{nH} ||v^*_s, b - v^*_s, b|| \leq \epsilon \cdot d \cdot \sqrt{m} + \epsilon \cdot d \cdot \sqrt{mn} \cdot \sqrt{nH} ||v^*_s, b - v^*_s, b|| \leq \epsilon d \sqrt{mn} ||v^*_s, b - v^*_s, b|| since $r_{s, b} \in [0, 1]^n$ and $v^*_s, b \in [0, H-1]^n$. $||v^*_s, b - v^*_s, b||$ is upper bounded by the maximum difference $\max_{s^2 \in S_{s^2}} (V(\pi^*, s^2) - V(\pi^*, s^2))$ over valuations in the next layer and $V(\pi^*, s^2) = V(\pi^*, s^2) = 0$ for $s^2 \in S_{s^2}$. Therefore, for any $s \in S$, recursive application of this computation gives the desired bound
\[V(\pi^*, s) - V(\pi^*, s) \leq (\epsilon d \sqrt{mn} (1 + \sqrt{nH})) H. \tag{43}\]

**References**

[1] H. Stackelberg, *Market Structure and Equilibrium*. New York, NY: Springer, 1934.
[2] D. Fudenberg, F. Drew, and D. K. Levine, *The Theory of Learning in Games*, vol. 2. Cambridge, MA, USA: MIT Press, 1998.
[3] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus, “Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games,” in *Proc. 7th Int. Joint Conf. Auton. Agents MultiAgent Syst.*, 2008, pp. 895–902.
[4] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, “Stackelberg security games: Looking beyond a decade of success,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 4944–5001.
[5] M. Tambe, *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[6] R. Yang, B. Ford, M. Tambe, and A. Lermieux, “Adaptive resource allocation for wildlife protection against illegal poachers,” in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst.*, 2014, pp. 453–460.
[7] M.-F. Balcan, A. Blum, N. Haghitalab, and A. D. Procaccia, “Commitment without regrets: Online learning in Stackelberg security games,” in *Proc. 16th ACM Conf. Econ. Comput. Comput.*, 2018, pp. 61–78.
[8] V. DeMiguel and H. Xu, “A stochastic multiple-leader Stackelberg model: Analysis, computation, and application,” *Oper. Res.*, vol. 57, no. 5, pp. 1220–1235, 2009.
[9] J. Letchford, V. Conitzer, and K. Munagala, “Learning and approximating the optimal strategy to commit to,” in *Algorithmic Game Theory*, M. Mavronicolas and V. G. Papadopoulou, Eds. Berlin, Germany: Springer, 2009, pp. 250–262.
[10] J. Marecki, G. Tesauro, and R. Segal. “Playing repeated Stackelberg games with unknown opponents,” in *Proc. 11th Int. Conf. Auton. Agents Multiagent Syst.*, 2012, vol. 2, pp. 821–828.
[11] A. Blum, N. Haghitalab, and A. D. Procaccia, “Learning optimal commitment to overcome insecurity,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1826–1834.
[12] A. Neyman, S. Sorin, and S. Sorin, *Stochastic Games and Applications*, vol. 570. Berlin, Germany: Springer, 2003.
[13] C.-Y. Wei, Y.-T. Hong, and C.-J. Lu, “Online reinforcement learning in stochastic games,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4994–5004.
[14] Y. Ouyang, H. Tavafoghi, and D. Teneketzis, “Dynamic games with asymmetric information: Common information based perfect bayesian equilibria and sequential decomposition,” *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 222–237, Jan. 2017.
[15] T. Li and S. P. Sethi, “A review of dynamic Stackelberg game models,” *Discrete Continuous Dyn. Syst. B*, vol. 22, no. 1, pp. 125–159, 2017.
[16] C. Chen and J. Cruz, “Stackelberg solution for two-person games with biased information patterns,” *IEEE Trans. Autom. Control*, vol. AC-17, no. 6, pp. 791–798, Jun. 1972.
[17] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.
[18] G. Neu, A. György, C. Szepesvari, and A. Antos, “Online Markov decision processes under bandit feedback,” *IEEE Trans. Autom. Control*, vol. 59, no. 6, pp. 676–691, Mar. 2014.
[19] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[20] B. von Stengel and S. Zamir, “Leadership with commitment to mixed strategies,” CDAM research report series (LSE-CDAM-2004-01), Centre Discrete Applicable Math., London School Econ. Political Sci., London, U.K., 2004.
[21] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, “Provably efficient reinforcement learning with linear function approximation,” in *Proc. 33rd Conf. Learn. Theory*, 2020, pp. 2137–2143.
[22] D. Silver, R. Sutton, and M. Müller, “Reinforcement learning of local shape in the game of Go,” in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 1053–1058.
[23] G. Neu and I. Olikovskaya, “Online learning in MDPs with linear function approximation and bandit feedback,” *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 10407–10417, 2021.
[24] D. Korzhyk, V. Conitzer, and R. Parr, “Complexity of computing optimal Stackelberg strategies in security resource allocation games,” in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 805–810.
[25] J. Abernethy and A. Rakhlin, “Beating the adaptive bandit with high probability,” in *Proc. IEEE Int. Theory Appl. Workshop*, 2009, pp. 280–289.
[26] G. Neu, “Explore no more: Improved high-probability regret bounds for non-stochastic bandits,” *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.
[27] M. Ghasemi, A. Hashemi, H. Vikalo, and U. Topcu, “No-regret learning with high-probability in adversarial Markov decision processes,” in Proc. 37th Conf. Uncertainty Artif. Intell., 2021, pp. 1–14.

[28] F. Chung and L. Lu, “Concentration inequalities and martingale inequalities: A survey,” Internet Math., vol. 3, no. 1, pp. 79–127, 2006.

[29] M. Ghasemi, A. Hashemi, H. Vikalo, and U. Topcu, “Online learning with implicit exploration in episodic Markov decision processes,” in Proc. IEEE Amer. Control Conf., 2021, pp. 1953–1958.

[30] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, “Gambling in a rigged casino: The adversarial multi-armed bandit problem,” in Proc. IEEE 36th Annu. Found. Comput. Sci., 1995, pp. 322–331.

[31] Gurobi Optimizer Reference Manual, Gurobi Optimization, LLC, Beaver-ton, OR, USA, 2021.

[32] I. Alabdulmohsin, X. Gao, and X. Zhang, “Efficient active learning of halfspaces via query synthesis,” in Proc. 29th AAAI Conf. Artif. Intell., 2015, pp. 2483–2489.

Niklas Lauffer received the B.S. degree in computer science and mathematics from the University of Texas at Austin, Austin, TX, USA, in 2021. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA. His research interests include developing autonomous learning and decision-making systems that are provably safe and beneficial by employing ideas from game theory, formal methods, and learning theory.

Mahsa Ghasemi received the B.Sc. degree in mechanical engineering from the Sharif University of Technology, Tehran, Iran, and the M.S.E. degree in mechanical engineering and the Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA. She is currently an Assistant Professor with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. Her research interests include theoretical and foundational advancements preparing autonomous systems to coexist with humans in our complex world, in particular, enabling efficient and reliable integration of autonomy in various applications, such as robotics, shared autonomy, and networked systems.

Abolfazl Hashemi (Member, IEEE) received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2014, and the M.S.E. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 2016 and 2020, respectively.

From 2020 to 2021, he was a Postdoctoral Scholar with the Oden Institute for Computational Engineering and Sciences, University of Texas at Austin. Since August 2021, he has been an Assistant Professor with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His research interests include optimization for machine learning, signal processing, and control.

Yagiz Savas received the B.Sc. degree in mechanical engineering from Bogazici University, Istanbul, Türkiye, in 2017. He is currently working toward the Ph.D. degree with the Department of Aerospace Engineering, University of Texas at Austin, Austin, TX, USA. His research interests include developing socially intelligent autonomous systems that coexist, cooperate, and compete with each other, as well as with humans, by drawing novel connections between controls, formal methods, and information theory.

Ufuk Topcu received the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2008.

He is currently an Associate Professor with the Department of Aerospace Engineering and Engineering Mechanics and the Oden Institute, University of Texas at Austin, Austin, TX, USA. His research interests include the theoretical, algorithmic, and computational aspects of design and verification of autonomous systems through novel connections between formal methods, learning theory, and controls.