LEARNING STACKELBERG EQUILIBRIA AND APPLICATIONS TO ECONOMIC DESIGN GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the use of reinforcement learning to learn the optimal leader's strategy in Stackelberg games. Learning a leader's strategy has an innate stationarity problem—when optimizing the leader's strategy, the followers' strategies might shift. To circumvent this problem, we model the followers via no-regret dynamics to converge to a Bayesian Coarse-Correlated Equilibrium (B-CCE) of the game induced by the leader. We then embed the followers' no-regret dynamics in the leader's learning environment, which allows us to formulate our learning problem as a standard POMDP. We prove that the optimal policy of this POMDP achieves the same utility as the optimal leader's strategy in our Stackelberg game. We solve this POMDP using actor-critic methods, where the critic is given access to the joint information of all the agents. Finally, we show that our methods are able to learn optimal leader strategies in a variety of settings of increasing complexity, including indirect mechanisms where the leader's strategy is setting up the mechanism's rules.

1 INTRODUCTION

In many economic settings, there's an inherent asymmetry between two or more participants—in market settings buyers are price takers, while sellers post a price, which can be dynamically updated by the sellers according observed changes in supply and demand. In matching mechanisms (school choice, residency assignment, etc.) the mechanism takes input from both sides in some form (e.g., truncated preference lists) and matches using a predetermined rule (e.g., running the Gale-Shapley algorithm). In these settings, and many others, one of the participants, *the leader*, has power to commit to a strategy (a seller or the matching mechanism), while others, *the followers* (buyers or participants in matching settings), respond to the strategy committed to by the leader. These problems are captured by the concept of *Stackelberg games*, where the leader has commitment power, and the followers reach an equilibrium with respect to the game induced by the leader's strategy. Stackelberg games have been extensively studied in the computer science literature, with applications to security (Sinha et al., 2018), wildlife conservation (Fang et al., 2016), and taxation policies (Zhou et al., 2019; Wang, 1999) among others.

While several works suggested an optimization-based approaches to computing Stackelberg equilibria (Vorobeychik & Singh, 2012; Conitzer & Sandholm, 2006; Sabbadin & Viet, 2016; Goktas & Greenwald, 2021; Basilico et al., 2017, e.g.), a more recent line of papers has shifted to using learning methods, which allow for the follower behavior to arise through learning dynamics rather than as the solution to an optimization problem. In particular, a major line of work has focused on normal form games (Letchford et al., 2009; Blum et al., 2014; Peng et al., 2019; Fiez et al., 2019, e.g.). However, normal form games can only be used to model a restricted set of leader-followers interactions. In our work, we consider more general settings where followers have private types and interactions with the leader occur across multiple rounds. We support a complex strategy space for the leader, where each strategy corresponds to a policy; in our most elaborated settings, the leader's problem is one of mechanism design, inducing a game with emergent communication between followers and the leader and where the leader adaptively chooses the order of play for followers and the prices they see.[1] Mguni et al. (2019); Cheng et al. (2017); Shi et al. (2020) and Shu & Tian (2019) also consider more general leader-followers interactions, but with few convergence results. To our knowledge, this work is the first one to provide a framework that supports a proof of convergence to a Stackelberg policy in multi-stage stochastic games.[2]

The main challenge with optimizing the leader's strategy in a Stackelberg game is that one needs to take into account the adaptation of follower strategies. In particular, we choose in this paper to model the followers as *no-regret learners*. Our crucial insight is the following: *Since followers compute their strategies by querying the leader's policy, this allows us to formulate these no-regret dynamics as a part of an extended POMDP (the "Stackelberg POMDP"), to be solved by the designer, where the policy's actions determine both the equilibrium play of followers and the reward to the leader. We prove that the optimal policy of this extended POMDP forms a Stackelberg equilibrium.* This approach allows us to model a population of followers with types sampled from a distribution, and to support highly complex, and adaptive, leader policies.

We demonstrate the robustness and flexibility of this algorithmic framework by demonstrating success in finding optimal leader strategies across a wide variety of settings of increasing complexity: (a) *Normal form games*, where the leader chooses a row and the follower chooses a column; (b) *Matrix design games*, where the leader modifies payoffs of a payoff matrix to impose a certain dominant strategy equilibrium; (c) *Message-allocation mechanisms*, where the leader assigns an item to the follower based on the follower's message to the leader; and (d) *Sequential price mechanisms with messages*, where the leader sets the rule of a sequential price mechanism based on an initial communication from the followers. The latter setting includes many of the common aspects of indirect mechanisms, such as limited communication and turn-taking play by the agents.[3] A detailed description of each of these experimental settings appears in Section 2.1.

For training, we use a modified actor-critic approach, where the actor does not have access to the internal state of the agents while the critic is given access to the full POMDP state, as is required for the critic to correctly evaluate the current state. This approach is based on the paradigm of *centralized training and decentralized execution* (CTDE), and is similar to the one proposed for DDPG (Lowe et al., 2017). Our experiments show that this use of CTDE leads to better performance than the use of an unmodified actor-critic algorithm.

2 Preliminaries

Setup. In a *Stackelberg game*, there is a leader with a strategy space $\mathcal{L}$ (which, in general, may include randomized strategies), and n followers, each with strategy spaces $\mathcal{F} = \mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_n$. There is a payoff function, $P : \mathcal{L} \times \mathcal{F} \to \Re^{n+1}_{\geq 0}$, mapping the joint strategy profile into payoffs for each of the leader and followers. We denote by P_0 as the leader's component in P, and by P_i, $i \in [n]$, as the ith follower's component in P. A Stackelberg game gives the leader the power to first commit to a strategy $\ell \in \mathcal{L}$. Subsequently, the followers play an equilibrium in the game defined by the induced payoff function $P(\ell, \cdot)$.[4] Let $\mathsf{EQ}(\ell) \in \mathcal{F}$ be a function that takes a leader's action and computes a resulting equilibrium (possibly in a randomized way). The goal of the leader is to play ℓ that maximizes $P_0(\ell, \mathsf{EQ}(\ell))$.

We also consider a Bayesian extension, where followers have payoff *types*, $\mathbf{t} = (t_1, \dots, t_n)$, sampled from a possibly correlated type distribution $\mathcal{D}$. The payoff function is accordingly augmented as $P(\mathbf{t}, \ell, \mathbf{a})$ for $\mathbf{a} \in \mathcal{F}$ and $\ell \in \mathcal{L}$, and similarly for $\mathsf{EQ}(\ell, \mathbf{t})$. The goal of the leader is also adjusted, and becomes that of choosing ℓ to maximize $\mathbb{E}_{\mathbf{t}\sim\mathcal{D}}[P_0(\ell, \mathbf{t}, \mathsf{EQ}(\ell, \mathbf{t}))]$. We sometimes consider cases where a follower's payoff is only a function of their own type and the joint actions and omit other followers' types in an individual follower payoff component.

[1] Bai et al. (2021) also consider a game with multiple rounds of interaction, but their methods, which are based on bandits approaches, are only practical when the leader's action space is finite and relatively small, and do not support Bayesian settings.

[2] Zhong et al. (2021) recently gave algorithms that support convergence for Markov games, but they restrict their followers to be myopic, and not to maximize their payoff in the game.

[3] An indirect mechanism is one in which agents communicate by sending messages that report less than their complete private information, for example sending a '0' or a '1', or selecting an item to purchase given prices.

[4] We will define our equilibrium concept in Definition 2.

Algorithm 1 Iterative No Regret Dynamics (Hartline et al., 2015)

---

parameters: number of iterations T
repeat T **times**
  Sample type profile $\mathbf{t} = (t_1, \ldots, t_n)$ from distribution $\mathcal{D}$
  Each agent i simultaneously and independently chooses an action $a_i \in \mathcal{A}_i$ using a no-regret algorithm that is specific for type t_i
  Each agent i obtains payoff $P_i(t_i, \mathbf{a})$, and updates their strategy, feeding the no-regret algorithm $P_i(t_i, (a'_i, \mathbf{a}_{-i}))$ for every $a'_i \in \mathcal{A}_i$.
end

Remark: In the rest of the section (Definitions 1, 2, Proposition 2.1 and Algorithm 1), we give general definitions and claims for games without a leader and omit the leader's strategy ℓ from the notation. In our context, we use these definitions to give the equilibrium concept, arising from the convergence dynamics that the followers follow in the game induced by the leader's action.

Bayesian coarse correlated equilibrium. The equilibrium concept that we adopt in modeling the followers' strategies is the coarse-correlated equilibrium and its Bayesian extension (as proposed by Hartline et al. (2015) for games of incomplete information).

Definition 1 (Coarse correlated equilibrium) *Consider a game with n agents, each with action space $\mathcal{A}_i$ ($\mathcal{A} = \times \mathcal{A}_i$), and a payoff function $P_i : \mathcal{A} \to \mathbb{R}$. Let σ be a joint randomized strategy profile from which the actions are sampled. σ is a* Coarse-Correlated Equilibrium (CCE) *if for every $a'_i \in \mathcal{A}_i$, we have $\mathbb{E}_{\mathbf{a}\sim\sigma}[P_i(\mathbf{a})] \geq \mathbb{E}_{\mathbf{a}\sim\sigma}[P_i(a'_i, \mathbf{a}_{-i})]$.*

Definition 2 (Bayesian coarse correlated equilibrium) *Consider a Bayesian game with n agents, each with type space $\mathcal{T}_i$ ($\mathcal{T} = \times \mathcal{T}_i$), action space $\mathcal{A}_i$ ($\mathcal{A} = \times \mathcal{A}_i$), and a payoff function $P_i : \mathcal{T}_i \times \mathcal{A} \to \mathbb{R}$. Let $\mathcal{D}$ be the joint type distribution of the agents, and $\mathcal{D}|_{t_i}$ the type distribution of all agents but i, conditioned on agent i having type t_i. Let $\sigma : \mathcal{T} \to \mathcal{A}$ be a joint randomized strategy profile that maps the type profile of all agents to an action profile. σ is a* Bayesian Coarse-Correlated Equilibrium (B-CCE) *if for every $a_i \in \mathcal{A}_i$, and for every $t_i \in \mathcal{T}_i$, we have*

$$\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i(\sigma(\mathbf{t}), t_i)] \geq \mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i((a_i, \sigma(\mathbf{t})_{-i}), t_i)],$$

where $\sigma(\mathbf{t})_{-i}$ is a mapping from type profile to action, excluding agent i's action.

In the above, agents jointly map their realized types to actions. This concept generalizes many other known solution concepts such as a Mixed Nash Equilibrium and a Bayes-Nash Equilibrium (BNE), where $\sigma = \prod \sigma_i$ and $\sigma(\mathbf{t}) = \prod \sigma_i(t_i)$, respectfully; that is, each agent chooses an action using a separate strategy σ_i, without correlating the action with other agents. These coarse correlated solution concepts are of interest because they arise naturally from agents iteratively updating their strategies using no-regret dynamics. This was observed for complete information games and CCE, and was extended for incomplete information games with independent type distributions and B-CCE (Hartline et al., 2015); in Proposition 2.1, we give a more general convergence result than Hartline et al. (2015) that holds for arbitrary, possibly correlated, type distributions.

Iterative no-regret dynamic. When updating agents' strategies, we invoke an iterative no-regret dynamics for Bayesian games, as described by Hartline et al. (2015). In these dynamics, each agent independently runs a no-regret algorithm for each of the types that the agent might have. Roughly speaking, at each round, a type profile $\mathbf{t} = (t_1, \ldots, t_n)$ of agents is sampled from $\mathcal{D}$. Then agents simultaneously choose their actions using the strategy associated with the sampled type t_i. After observing all other agents' actions, each agent then privately uses a no-regret algorithm to update their strategy for their own type t_i. We provide further details in Algorithm 1. In Appendix B, we show that Iterative No Regret Dynamics converges to a B-CCE for any type distribution, which is a more general convergence result than the one shown in Hartline et al. (2015).

Proposition 2.1 *For every type distribution $\mathcal{D}$, the empirical distribution over the history of actions in the iterative no regret dynamics algorithm for any type profile $\mathbf{t}$ converges to a B-CCE. That is, for every $\mathcal{D}$, i, t_i, and a_i,*

$$\lim_{T\to\infty} \quad \mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i(\sigma^{\text{emp}}(\mathbf{t}), t_i)] \; - \; \mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i((a_i, \sigma^{\text{emp}}(\mathbf{t})_{-i}), t_i)] \geq 0,$$

Setting	Followers	Bayesian	Leader Strategy
Normal form games	Single	No	Choose a row
Matrix design games	Multiple	No	Modify table payoffs
Simple allocation mechanism	Single	Yes	Map message to allocation
μSPM	Multiple	Yes	Map messages to an SPM

Table 1: A taxonomy of the different experimental settings that we study.

where σ^{emp} is the empirical distribution after T steps.

The specific no-regret algorithm that we use for updating agents' strategies is the *multiplicative-weights algorithm* (see Appendix D).

In the paper, we present a POMDP formulation whose optimal policy gives asymptotic convergence to Stackelberg Equilibrium where the followers form an approximate B-CCE. In order to define what *asymptotic convergence* means, we introduce the following definitions.

Definition 3 (ϵ-Approximate B-CCE) *A joint strategy profile σ is an ϵ-approximate B-CCE for $\epsilon > 0$ if, for every $a_i \in \mathcal{A}_i$, and for every $t_i \in \mathcal{T}_i$, we have*

$$\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i(\sigma(\mathbf{t}), t_i)] \geq \mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i((a_i, \sigma(\mathbf{t})_{-i}), t_i)] - \epsilon,$$

where $\sigma(\mathbf{t})_{-i}$ is a mapping from type profile to action, excluding agent i's action.

An alternative way to frame Proposition 2.1 is that for every ϵ there exists $T(\epsilon)$ such that for every $T' \geq T$, the empirical distribution of the T'-rounds no-regret dynamics is an ϵ-approximate BCCE in expectation.

Definition 4 (ϵ-Approximate Optimal Leader Strategy) *Let $\epsilon\text{-}\mathsf{EQ}(\ell, \mathbf{t})$ be a function that takes leader strategy ℓ and followers' types $\mathbf{t}$ and returns an ϵ-approximate B-CCE for some $\epsilon > 0$. The goal of the leader is to choose ℓ to maximize*

$$\mathbb{E}_{\mathbf{t}\sim\mathcal{D}}[P_0(\ell, \mathbf{t}, \epsilon\text{-}\mathsf{EQ}(\ell, \mathbf{t})].$$

2.1 Stackelberg Settings

Going forward, when referring to a Stackelberg setting or Stackelberg equilibrium we mean in a CCE or B-CCE of the game that is induced by a leader's strategy. In the following, we cover the Stackelberg settings that appear in our experimental results (see also Table 1).

Normal form games. In a normal form game, the agent payoffs are specified via a matrix. The leader's strategy is to choose a row, and as a response the follower chooses a column. In the randomized variant, the leader's strategy space is the set of distributions over the rows. Notice that in both cases, the follower's optimal strategy is deterministic; i.e., to choose a single row.

Matrix design game. In a matrix design game, and inspired by Monderer & Tennenholtz (2003), the leader is given a game matrix to be played by the followers. The leader can then modify the payoff matrix by changing some of the payoffs in the matrix by a positive amount. The followers then choose an equilibrium in the matrix with the modified matrix game. Crucially, the leader's goal is to maximize followers' welfare without actually making monetary transfers in equilibrium (i.e., the leader modifies payoffs, in a way where the followers do not choose, in equilibrium, a matrix cell where the payoff was changed).

Simple allocation mechanism. In this type of mechanism, a leader needs to allocate one of k items to a single follower who is interested in only one of them. When the game begins, the follower's item of interest (i.e., their type) is sampled uniformly at random. The follower observes their item of interest and sends a message $\mu \in \{1, .., m\}$ to the leader, and the leader allocates one of the items to the follower. If the follower receives their item of interest, the utility of both agents is 1, and 0 otherwise. Importantly, the number of messages available to the follower m can be much smaller than k.

Sequential price mechanisms with messages. Here we consider the family of sequential price mechanisms as studied in Brero et al. (2021), extended here to include an additional messaging round similar to the one used for our simple allocation mechanisms. There is some set of indivisible, distinct items to allocate to a set of agents. In the first round, each agent i simultaneously sends a *message* $\mu_i \in \{1, .., m\}$ to the mechanism, for some choice of $m > 1$. The mechanism then visits each agent in turn in each of the following rounds, possibly in some adaptive order. In each of these rounds, the mechanism picks an unvisited agent i, and posts a price p_j for each item j that remains available. Then, agent i picks the bundle of items that maximizes their utility, and is charged accordingly. The goal of the mechanism is to maximize the expected social welfare. In this setting, the leader's strategy is to pick a μSPM , and the followers respond by choosing their messaging strategy.

3 FORMULATION AS A SINGLE AGENT PROBLEM

In this section, we show how to formulate the problem of finding an optimal leader strategy—which, in its straightforward form involves multiple agents—as a single-agent problem. Our *Stackelberg POMDP* (Section 4) builds on this single-agent formulation. For concreteness, we instantiate the formulation to the μSPM setting, which is the most complicated setting we investigate.

Consider a *partially observable stochastic game* (Hansen et al., 2004) among the n followers and a leader. The game state, s^τ, in round τ is a tuple consisting of the type profile $\mathbf{t}$, the (initially empty) action profile $\mathbf{a}$, and the other parameters required to represent the specific Stackelberg setting at hand. For instance, for μSPM, we keep the current partial allocation, and the residual setting consisting of agents not yet visited and items not yet allocated.

In round 0, each follower observes their own type and picks an action a_i simultaneously with other followers, and according to an equilibrium strategy induced by the leader. In round 1, the leader observes the followers' actions. In any round $\tau \geq 1$, the leader takes action a^τ. The set of actions the leader takes represents the leader's strategy.

State transitions are deterministic. The first state transition simply consists of adding followers' actions to the state. Then, in any round $\tau > 0$, the state $s^{\tau+1}$ is obtained by changing the game's parameters according to the leader's action at the previous state. For the μSPM setting, it is obtained by adding the bundle selected by agent to their allocation to form a new partial allocation, and the items and agent are removed from the residual setting.

Since the followers' strategies are a function of the leader's policy, we can write the optimization problem as an optimization over the policy's parameters θ. Let σ_θ an equilibrium induced by a policy represented by parameters θ. Let $\mathsf{tr} = (s^0, a^0, s^1, a^1, \dots, s^T, a^T)$ be a trajectory of an episode of the stochastic game. Our objective is to find θ that maximizes

$$J(\theta) = E_{\mathsf{tr}\sim p_\theta}\left[\sum_{\tau=0}^{T} r(s^\tau, a^\tau)\right], \text{ where} \tag{1}$$

$$p_\theta(\mathsf{tr}) = p_\theta(s^0)\prod_{\tau=0}^{T}\pi_\theta(a^\tau|o^\tau)p(s^{\tau+1}|s^\tau, a^\tau) = p(\mathbf{t})\,\sigma_\theta(\mathbf{a} \mid \mathbf{t})\prod_{\tau=0}^{T}\pi_\theta(a^\tau|o^\tau)p(s^{\tau+1}|s^\tau, a^\tau). \tag{2}$$

Observation 3.1 *Consider θ^* that maximizes Equation 1, then when the leader adopts this policy, and the followers use strategies σ_{θ^*}, the tuple $(\theta^*, \sigma_{\theta^*})$ forms a Bayesian Stackelberg equilibrium.*

This follows immediately, since each follower best responds to the policy and other followers' strategies, and by the optimality of θ^*, there is no policy that could achieve better by switching to θ' and and letting followers play according to the new equilibrium $\sigma_{\theta'}$.

To model followers' strategies, we use *no-regret dynamics* (via multiplicative-weights, MW) to compute the equilibrium of the game induced by a particular leader's policy with parameters θ. We wish to optimize the leader's policy π_θ using *policy gradient* methods. The reason we cannot directly differentiate the objective is that the follower actions depend on their strategies, which means that we need to differentiate follower strategies as a result of the policy change.

The crucial insight is that we can now expand σ_θ to be the initial part of a POMDP that calls the policy multiple times to compute followers' strategies through no-regret dynamics. Note that in no-regret dynamics, followers update their strategies by accessing their payoff vector produced by examining their payoffs for the outcome produced by the policy for different possible actions they might take; therefore, this can actually be computed by rolling out the policy itself. By doing this, we can apply RL methods to optimize over the parameters of the policy that solves this POMDP. In the next section, we give the POMDP formulation of this problem. This gives rise to a single agent RL problem that we can solve using standard gradient descent methods (as per Appendix A, where we provide an actor-critic approach).

4 STACKELBERG POMDP

Our POMDP, which we name the *Stackelberg POMDP*, has a long episode that consists of multiple sub-episodes. A sub-episode consists of rolling out the policy to determine a payoff for the followers. We have two types of sub-episodes:

1. *Equilibrium sub-episodes:* The first set of sub-episodes are used by the followers to find the equilibrium of the game given the current leader policy, where for some T rounds of no-regret dynamics:
   (a) Followers' types are drawn (this is meaningful only in Bayesian settings),
   (b) Followers jointly sample their actions according to their current strategy,
   (c) Each follower runs an algorithm that minimizes its external-regret[5] where the no-regret algorithm has access to all possible payoffs[6]: it computes the payoff it would have received for every possible message sent, fixing all other agents' messages. The agent then uses this payoff vector to update its strategy using a no-regret algorithm.
2. *Reward sub-episode:* After running T equilibrium sub-episodes, we run one or more sub-episodes where the policy gets a reward for followers playing the current messaging strategy (as determined in the first T rounds).

We note that this is a POMDP since the policy is not aware of the followers' strategies, whether we are in an equilibrium sub-episode or a reward sub-episode, and in Bayesian settings, the followers' private types which are all part of the state of the POMDP.

In Appendix C, we give a full description of the Stackelberg POMDP. The following proposition shows that the optimal solution of the Stackelberg POMDP gives the desired result.

Definition 5 (Asymptotic convergence to optimal leader strategy) *We say the Stackelberg POMDP achieves an asymptotic convergence to optimal leader strategy if for every ϵ there exists $T = T(\epsilon)$ such that the following holds. For every $T' > T$, if the Stackelberg POMDP uses T' equlibrium sub-episodes, then the optimal policy for the POMDP is an ϵ-approximate optimal leader strategy in expectation (Definition 4).*

Proposition 4.1 *The Stackelberg POMDP achieves an asymptotic convergence to an optimal leader strategy.*

Proof 4.1 *Fix $\epsilon > 0$. By Proposition 2.1, there exists some $T = T(\epsilon)$ for which if we run no-regret-dynamics for T rounds, then the resulting followers' strategies an ϵ-approximate B-CCE. Let θ^*_T be the optimal policy for the Stackelberg POMDP for T first phase steps, and $\sigma_{\theta^*_T}$ the strategies implied by running the first phase with policy θ^*_T for T time steps. Since followers play an ϵ-approximate B-CCE with respect to leader's strategy, the leader strategy is an ϵ-approximate optimal leader strategy and the Stackelberg POMDP achieves an asymptotic convergence to an optimal leader strategy.*

[5]External regret compares the performance of a sequence of actions to the performance of the best single action in hindsight.

[6]This can be changed to an internal regret minimization in the case where the no-regret algorithm only has access to the payoff of the actual joint actions chosen by the agents. In this case, the algorithm will need more steps to get convergence.

Observation 4.1 *Our Stackelberg POMDP models followers to use no-regret-dynamics and the solution concept we get to is a Stackelberg Equilibrium where the followers' equilibrium is a Bayesian CCE. Since we deal with Bayesian settings where agents have private types, a Bayesian solution concept is appropriate. When our methods are applied to non-Bayesian settings, then the solution concept would change to the non-Bayesian counterpart (i.e., in the normal-form games and matrix design games scenarios). If the agents use "no swap regret" to update their strategy instead of "no external regret", then the solution concept would further change from coarse correlated equilibrium to the widely studied correlated equilibrium solution concept. The reason we chose to model agents as no external regret agents is because no external regret algorithms have stronger convergence guarantees than their no-swap regret counterparts. More generally, we note that our Stackelberg POMDP is well formed as long as the re-equilibration behavior of the sellers can be modeled through Markovian dynamics and integrated into the Stackelberg POMDP states. For example, one can imagine using other common methods to derive sellers' response strategies such as Q-learning—as already successfully experimented in a follow-up work (Brero et al., 2022)—or other reinforcement learning approaches.*

5 Experimental Results

In this section, we demonstrate the robustness and flexibility of our learning approach via different experiments. The set of settings we consider is described in Section 2.1.

We train the platform policy with MAPPO, which is a multi-agent variant of the *Proximal Policy Optimization* (PPO) algorithm described by Schulman et al. (2017). To implement MAPPO, we start from the PPO version implemented in *Stable Baselines3* (Raffin et al., 2021, MIT License) and modify it to let the critic network access the followers' internal information.[7] In the remainder of this section, we use standard PPO as a baseline and compare its training performance with MAPPO.

We train for 10 million steps, unless otherwise specified. We log rewards in the reward phase of a separate Stackelberg POMDP episode that we run every 10k training steps. These evaluation episodes use the current leader policy, and operate it executing the action with the highest weight given each observation.

The number of equilibrium steps in our Stackelberg POMDPs is always large enough to guarantee convergence of the followers' MW dynamics. This means 100 steps in the matrix experiments of Section 5.1 and Section 5.2, and 1000 steps in the economic design experiments of Section 5.3 and Section 5.4. The larger number of equilibrium steps in our economic design experiments is due to the fact that we sample followers' private types at the beginning of each game. This also motivates a larger number of reward steps (100) in our economic design experiments compared to the ones used in the matrix ones (10).

5.1 Normal Form Games

We consider the second matrix game described in Zhang et al. (2019) (see Table 2).[8] Here, the optimal deterministic leader action is A, with the follower responding with action A and generating payoff 20 to the leader. However, the leader can further improve her payoff by picking a randomized action. Indeed, they can play A with probability $0.25 + \varepsilon$ and B with probability $0.75 - \varepsilon$. In this scenario, the follower is still maximizing his payoff by playing A and the leader's expected utility is $27.5 + \varepsilon$. We then also consider the scenario where the leader can randomize over rows. For this, we formulate a POMDP where the leader actions are weights over rows, and derive randomized game actions by normalizing these weights. The results are given in Figure 1. MAPPO immediately learns to play the optimal strategy when actions are deterministic. At the same time, the randomized scenario requires a short learning phase before reaching the optimal performance after around 100k training steps. Similar results can be achieved by running PPO instead of MAPPO.

[7] Furthermore, to reduce variance in followers' responses (due to non-deterministic policy behavior of the leader), we maintain an *observation-action map* throughout each episode. When a new observation is encountered during the episode, the policy chooses an action following the default training behavior and stores this new observation-action pair in the map. Otherwise, the policy simply uses the action stored in the map.

[8] We present the results for the first matrix game described in Zhang et al. (2019) in Appendix E.

	A	B	C
A	$(20, 15)$	$(0, 0)$	$(0, 0)$
B	$(30, 0)$	$(10, 5)$	$(0, 0)$
C	$(0, 0)$	$(0, 0)$	$(5, 10)$

Table 2: Matrix game 2.

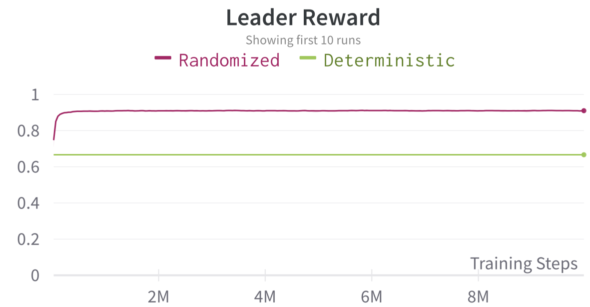


Figure 1: Training curves for matrix game in Table 2 averaged over 10 runs when the leader policy is trained using MAPPO. Rewards are normalized between 0 and 1.

	A	B
A	$(3 + \tau, 3)$	$(6, 4)$
B	$(4, 6)$	$(2, 2 + \tau)$

Table 3: Matrix game (Monderer & Tennenholtz, 2003)

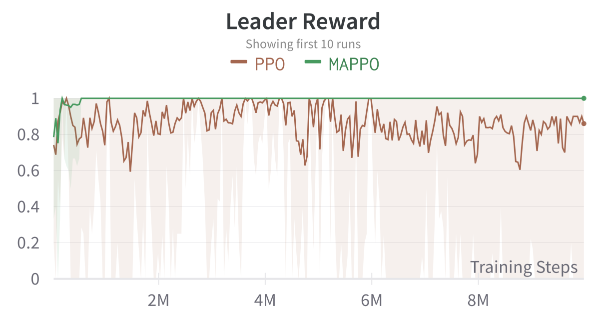


Figure 2: Training curve for matrix game in Table 3 averaged over 10 runs. Rewards are normalized between 0 and 1. Payment τ is chosen from set $\{0, 1, .., 10\}$.

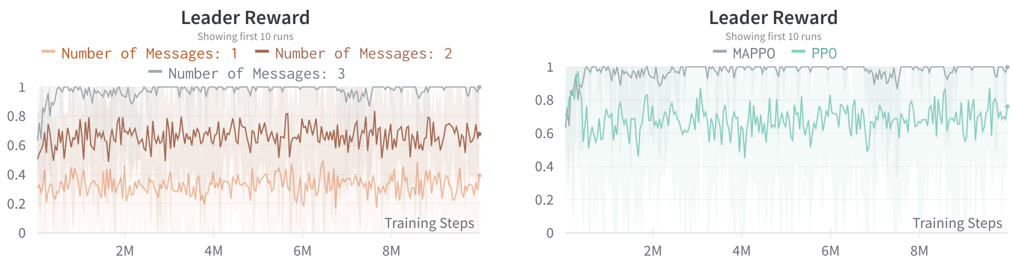
5.2 Matrix Design Game

We consider the first example presented by Monderer & Tennenholtz (2003). The followers' game is described in Table 3. We notice that, when the leader does not intervene (i.e., $\tau = 0$), the game has no dominant strategy equilibrium. At the same time, when $\tau \geq 4$, actions A and B become dominant strategies for the row and the column players, respectively. Following Monderer & Tennenholtz (2003), we assume that the leader's goal is to have the followers play different actions and for this we set the leader' reward to 1 when this is the case and 0 otherwise. The results are displayed in Figure 2. Here MAPPO significantly outperforms PPO, learning to play $\tau \geq 4$ after around 200k training steps. PPO has an oscillatory behavior and does not always provide optimal leader behavior. This is expected, since as the RL algorithm does not have access to the full state, which includes agents' strategies, the critic in PPO cannot correctly evaluate the current state of the MDP. This is addressed through the centralized critic of MAPPO.[9]

5.3 Simple Allocation Mechanisms

We consider the simple allocation mechanisms described in Section 2 with 3 items ($k = 3$). We start by testing the performance of MAPPO when varying m in $\{1, 2, 3\}$. The results are displayed in Figure 3a. When $m = 1$, the follower cannot inform the leader, which chooses a random item and succeeds only one third of the time. When $m = 2$, the follower can signal its item of interest one third of the time, allowing the leader to increase her expected reward to $0.\bar{6}$. When $m = 3$, the follower can signal his type, and the leader can always realize her maximum reward (i.e., 1). As we can notice, MAPPO always realizes this optimal payoffs after a relatively short training phase.

[9] See Appendix F for a more accurate analysis of the critic network performance.



(a) Message spaces comparison. **(b)** MAPPO vs. PPO, message space size $m = 3$.

Figure 3: Training curves for Simple Allocation Mechanisms, averaged over 10 runs.

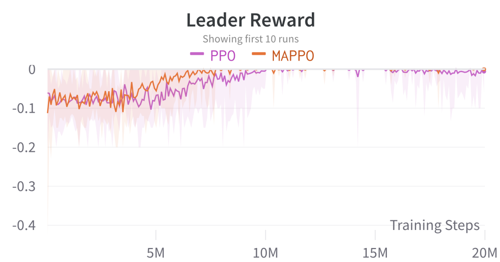


Figure 4: μSPM training curves averaged over 10 runs.

We also compare MAPPO and PPO in the scenario with $m = 3$. As we can see from Figure 3b, MAPPO significantly outperforms PPO.

5.4 Sequential Price Mechanisms with Messages

We consider the setting introduced by Agrawal et al. (2020) in their Example 1. We have two agents and one item. Each agent's type corresponds to his value for the item. Agent 1's type, t_1, has support $\{1/2, 1/(2\varepsilon)\}$ with probabilities $\{1 - \varepsilon, \varepsilon\}$, and agent 2's value, t_2, has support $\{0, 1\}$ with probabilities $\{1/2, 1/2\}$. In our experiments, we use $\varepsilon = 0.2$. The welfare-optimal allocation cannot always be realized when followers cannot signal their types.[10] However, there exists an μSPM with $m = 2$ in which agents are properly incentivized to communicate their types and where the allocation always optimizes welfare:

1. If agent 1 sends message 0, then the mechanism uses a minimal price (e.g., 0.1) in each round, and visits agent 2 first;
2. Otherwise, the mechanism uses a price between $1/2$ and $1/(2\varepsilon)$ in the first round and visits agent 1 first.

In the equilibrium, the optimal allocation is always achieved if agent 1 bids 0 when their value is low. At the same time, agent 1 is only motivated to bid 0 when their value is low, given that the price offered in (2.) would be too high in this case. As previously done by Brero et al. (2021), albeit in a setting without communication, we set the leader's reward to the welfare loss (and not welfare). To avoid exploding gradients, we reduce the default PPO learning rate by a factor of 100, and we train for a longer period (20M steps). As we can see from Figure 4, MAPPO reaches an optimal performance while PPO is still not stable at the end of our training run.

[10]The welfare-optimal sequential price mechanism without messaging visits agent 2 first and then agent 1, using price zero in both cases. Here, the welfare-optimal allocation is not implemented when $v_1 = 1/2$ and $v_2 = 1/(2\varepsilon)$.

Reproducibility Statement

All source code used for experiments will be submitted as part of the supplementary material, along with detailed instructions on how to recreate the experiments presented in this paper. We plan to release the source code of our experiments under an open-source license upon acceptance.

References

Shipra Agrawal, Jay Sethuraman, and Xingyu Zhang. On optimal ordering in the optimal stopping problem. In *Proc. EC '20: The 21st ACM Conference on Economics and Computation*, pp. 187–188, 2020.

Yu Bai, Chi Jin, Huan Wang, and Caiming Xiong. Sample-efficient learning of Stackelberg equilibria in general-sum games. In *Advances in Neural Information Processing Systems 34*, pp. 25799–25811, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d82118376df344b0010f53909b961db3-Abstract.html.

Nicola Basilico, Stefano Coniglio, and Nicola Gatti. Methods for finding leader–follower equilibria with multiple followers. *arXiv preprint arXiv:1707.02174*, 2017.

Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. *Advances in Neural Information Processing Systems*, 27, 2014.

Gianluca Brero, Alon Eden, Matthias Gerstgrasser, David C. Parkes, and Duncan Rheingans-Yoo. Reinforcement learning of sequential price mechanisms. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, pp. 5219–5227, 2021.

Gianluca Brero, Nicolas Lepore, Eric Mibuari, and David C. Parkes. Learning to mitigate AI collusion on economic platforms. *CoRR*, abs/2202.07106, 2022. URL https://arxiv.org/abs/2202.07106.

C. Cheng, Z. Zhu, B. Xin, and C Chen. A multi-agent reinforcement learning algorithm based on Stackelberg game. In *6th Data Driven Control and Learning Systems (DDCLS)*, pp. 727–732, 2017.

Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings 7th ACM Conference on Electronic Commerce (EC-2006)*, pp. 82–90, 2006.

Fei Fang, Thanh Hong Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, Andrew Lemieux, et al. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *AAAI*, volume 16, pp. 3966–3973, 2016.

Tanner Fiez, Benjamin Chasnov, and Lillian J Ratliff. Convergence of learning dynamics in Stackelberg games. *arXiv preprint arXiv:1906.01217*, 2019.

Denizalp Goktas and Amy Greenwald. Convex-concave min-max Stackelberg games. *Advances in Neural Information Processing Systems*, 34:2991–3003, 2021.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 709–715. AAAI Press / The MIT Press, 2004. URL http://www.aaai.org/Library/AAAI/2004/aaai04-112.php.

Jason Hartline, Vasilis Syrgkanis, and Eva Tardos. No-regret learning in bayesian games. *arXiv preprint arXiv:1507.00418*, 2015.

Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *International symposium on algorithmic game theory*, pp. 250–262. Springer, 2009.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30*, pp. 6379–6390, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html.

David Mguni, Joel Jennings, Emilio Sison, Sergio Valcarcel Macua, Sofia Ceppi, and Enrique Munoz de Cote. Coordinating the crowd: Inducing desirable equilibria in non-cooperative systems. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 386–394, 2019.

Dov Monderer and Moshe Tennenholtz. k-implementation. In *Proceedings of the 4th ACM conference on Electronic Commerce*, pp. 19–28, 2003.

Binghui Peng, Weiran Shen, Pingzhong Tang, and Song Zuo. Learning optimal strategies to commit to. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pp. 2149–2156, 2019.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Régis Sabbadin and Anne-France Viet. Leader-follower MDP models with factored state space and many followers. In *Proc. 22nd European Conference on Artificial Intelligence*, pp. 116–124, 2016.

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

Zhenyu Shi, Runsheng Yu, Xinrun Wang, Rundong Wang, Youzhi Zhang, Hanjiang Lai, and Bo An. Learning expensive coordination: An event-based deep RL approach. In *8th International Conference on Learning Representations*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=ryeG924twB`.

Tianmin Shu and Yuandong Tian. M^3rl: Mind-aware multi-agent management reinforcement learning. In *7th International Conference on Learning Representations*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=BkzeUiRcY7`.

Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. IJCAI, 2018.

Yevgeniy Vorobeychik and Satinder P. Singh. Computing Stackelberg equilibria in discounted stochastic games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

You-Qiang Wang. Commodity taxes under fiscal competition: Stackelberg equilibrium and optimality. *American Economic Review*, 89(4):974–981, 1999.

Haifeng Zhang, Weizhe Chen, Zeren Huang, Minne Li, Yaodong Yang, Weinan Zhang, and Jun Wang. Bi-level actor-critic for multi-agent coordination, 2019.

Han Zhong, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Can reinforcement learning find Stackelberg-Nash equilibria in general-sum markov games with myopic followers? *arXiv preprint arXiv:2112.13521*, 2021.

Dequn Zhou, Yunfei An, Donglan Zha, Fei Wu, and Qunwei Wang. Would an increasing block carbon tax be better? A comparative study within the Stackelberg game framework. *Journal of environmental management*, 235:328–341, 2019.

A AN ACTOR-CRITIC APPROACH

In Section 4, the entire optimization problem is formulated as a POMDP, where the only actions are the actions of the policy. Let $\mathsf{tr} = (s^0, a^0, \ldots, s^T, a^T)$ be a trajectory of the POMDP described above. We can express the trajectory probability of the optimization problem in equation 1 in a standard way

$$p_\theta(\mathsf{tr}) \quad = \quad p(s^0) \prod_{t=0}^{T} \pi_\theta(a^t|o^t) p(s^{t+1}|s^t, a^t),$$

where $p(s^0)$ now doesn't depend on θ, and the states and observations are defined by the Stackelberg POMDP described above. The gradient of $J(\theta)$ with respect to θ can be expressed as

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\mathsf{tr}\sim p_\theta}\left[\nabla_\theta \log p_\theta(\mathsf{tr}) \left(\sum_{t'=1}^{T} r(s^{t'}, a^{t'})\right)\right] \\
&= \mathbb{E}_{\mathsf{tr}\sim p_\theta}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a^t|o^t) \left(\sum_{t'=1}^{T} r(s^{t'}, a^{t'})\right)\right] \\
&= \mathbb{E}_{\mathsf{tr}\sim p_\theta}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a^t|o^t) \left(\sum_{t'=t}^{T} r(s^{t'}, a^{t'})\right)\right],
\end{aligned}$$

where the last equality follows since future actions do not affect past rewards in an POMDP. $\nabla_\theta J(\theta)$ is approximated by sampling ℓ different trajectories $\mathsf{tr}_1, .., \mathsf{tr}_\ell$:

$$\nabla_\theta J(\theta) \quad \approx \quad \frac{1}{\ell} \sum_{k=1}^{\ell} \sum_{t=1}^{T^\ell} \nabla_\theta \log \pi_\theta(a_k^t|o_k^t) \left(\sum_{t'=t}^{T^\ell} r(s_k^{t'}, a_k^{t'})\right).$$

One problem with this approach is that its gradient approximation has high variance as its value depends on sampled trajectories. A common solutionis to replace each term $\sum_{t'=t}^{T} r(s_k^{t'}, a_k^{t'})$ with $Q^\theta(s_k^t, a_k^t)$, where

$$Q^\theta(s, a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[r(s, a) + \gamma \mathbb{E}_{a'\sim\pi^\theta(s')}[Q^\theta(s', a')]] \tag{3}$$

is the "critic". Note that $Q^\theta(s, a)$ can be accessed at training time as we have access to the full state of the POMDP. This approach based on centralized training and decentralized execution is similar to the one proposed for DDPG by Lowe et al. (2017).

B CONVERGENCE TO EQUILIBRIUM

Hartline et al. (2015) consider using a no-regret dynamic on a fixed game of incomplete information. They show that, for independent valuations, the time-averaged history of agents' actions converges to a B-CCE (see Definition 2) if each agent follows a no-regret learning algorithm for each of her types. They assume independent type distributions for their analysis. We extend their theory to show that no-regret dynamics also converge to a B-CCE for arbitrary type distributions.

Proposition 2.1 *For every type distribution $\mathcal{D}$, the empirical distribution over the history of actions in the iterative no regret dynamics algorithm for any type profile $\mathbf{t}$ converges to a B-CCE. That is, for every $\mathcal{D}$, i, t_i, and a_i,*

$$\lim_{T\to\infty} \quad \mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i(\sigma^{\text{emp}}(\mathbf{t}), t_i)] \;-\; \mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}[P_i((a_i, \sigma^{\text{emp}}(\mathbf{t})_{-i}), t_i)] \geq 0,$$

where σ^{emp} is the empirical distribution after T steps.

Proof Fix agent i, and a type t_i. Let $\tau(t_i)$ denote the set of time steps at which t_i was sampled as the type of agent i, $\tau(\mathbf{t})$ denote the set of time steps at which $\mathbf{t}$ was sampled as the type profile, and $\tau(\mathbf{t}, \mathbf{a})$ denote the set of time steps at which $\mathbf{t}$ was sampled and $\mathbf{a}$ were the agents' actions. The expected value of agent i with type t_i when playing the agent's action according to the empirical distribution of actions σ^{emp} is:

$$\begin{aligned}
\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}\left[P_i(\sigma^{\text{emp}}(\mathbf{t}), t_i)\right] &= \sum_{\mathbf{t}_{-i}} \Pr[\mathbf{t}_{-i}] \sum_{\mathbf{a}} \Pr[\sigma^{\text{emp}}(\mathbf{t}) = \mathbf{a}] P_i(\mathbf{a}, t_i) \\
&= \sum_{\mathbf{t}_{-i}} \frac{|\tau(\mathbf{t})|}{|\tau(t_i)|} \sum_{\mathbf{a}} \Pr[\sigma^{\text{emp}}(\mathbf{t}) = \mathbf{a}] P_i(\mathbf{a}, t_i) \; + \\
&\quad \left(\Pr[\mathbf{t}_{-i}] - \frac{|\tau(\mathbf{t})|}{|\tau(t_i)|}\right) \sum_{\mathbf{a}} \Pr[\sigma^{\text{emp}}(\mathbf{t}) = \mathbf{a}] P_i(\mathbf{a}, t_i). \qquad (4)
\end{aligned}$$

Notice that by the Glivenko-Cantelli Theorem, as $T \to \infty$, $\left(\Pr[\mathbf{t}_{-i}] - \frac{|\tau(\mathbf{t})|}{|\tau(t_i)|}\right) \to 0$. Since it is multiplied by $\Pr[\sigma^{\text{emp}}(\mathbf{t}) = \mathbf{a}] P_i(\mathbf{a}, t_i) \leq \Pr[\sigma^{\text{emp}}(\mathbf{t}) = \mathbf{a}] P_i(\mathbf{a}, t_i)$, a bounded term, the second summand of Eq. (4) goes to 0 as $T \to \infty$.

As for the first summand, we have

$$\begin{aligned}
\sum_{\mathbf{t}_{-i}} \frac{|\tau(\mathbf{t})|}{|\tau(t_i)|} \sum_{\mathbf{a}} \Pr[\sigma^{\text{emp}}(\mathbf{t}) = \mathbf{a}] P_i(\mathbf{a}, t_i) &= \sum_{\mathbf{t}_{-i}} \frac{|\tau(\mathbf{t})|}{|\tau(t_i)|} \sum_{\mathbf{a}} \frac{|\tau(\mathbf{t}, \mathbf{a})|}{|\tau(\mathbf{t})|} P_i(\mathbf{a}, t_i) \\
&= \left(\sum_{\mathbf{t}_{-i}} \sum_{\mathbf{a}} |\tau(\mathbf{t}, \mathbf{a})| P_i(\mathbf{a}, t_i) \right) / |\tau(t_i)| \\
&= \left(\sum_{\mathbf{t}_{-i}} \sum_{\mathbf{a}} \sum_{\tau \in \tau(\mathbf{t})} P_i(\mathbf{a}, t_i) \cdot \mathbb{1}_{\mathbf{a}^t = \mathbf{a}} \right) / |\tau(t_i)| \\
&= \left(\sum_{\mathbf{t}_{-i}} \sum_{\tau \in \tau(\mathbf{t})} \sum_{\mathbf{a}} P_i(\mathbf{a}, t_i) \cdot \mathbb{1}_{\mathbf{a}^\tau = \mathbf{a}} \right) / |\tau(t_i)| \\
&= \left(\sum_{\mathbf{t}_{-i}} \sum_{\tau \in \tau(\mathbf{t})} P_i(\mathbf{a}^\tau, t_i) \right) / |\tau(t_i)| \\
&= \sum_{\tau \in \tau(t_i)} P_i(\mathbf{a}^\tau, t_i) / |\tau(t_i)|.
\end{aligned}$$

Therefore, we can write Eq. (4) as

$$\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}\left[P_i(\sigma^{\text{emp}}(\mathbf{t}), t_i)\right] = \sum_{\tau \in \tau(t_i)} P_i(\mathbf{a}^\tau, t_i) / |\tau(t_i)| \; + \; \alpha(T),$$

where $\alpha(T) \to 0$ as $T \to \infty$.

By replacing $\sigma^{\text{emp}}(\mathbf{t})_i$ with an arbitrary fixed action a'_i in the above derivation, we get that

$$\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}\left[P_i((a'_i, \sigma^{\text{emp}}(\mathbf{t})_{-i}), t_i)\right] = \sum_{\tau \in \tau(t_i)} P_i((a'_i, \mathbf{a}^\tau_{-i}), t_i) / |\tau(t_i)| \; + \; \beta(T),$$

where $\beta(T) \to 0$ as $T \to \infty$.

Since agent i uses a no-regret algorithm for each type t_i, we have that for $T \to \infty$ (which implies $|\tau(t_i)| \to \infty$),

$$\mathbb{E}_{\mathbf{a}^1, \ldots, \mathbf{a}^T}\left[\sum_{\tau \in \tau(t_i)} P_i(\mathbf{a}^\tau, t_i) / |\tau(t_i)| \right] \geq \mathbb{E}_{\mathbf{a}^1, \ldots, \mathbf{a}^T}\left[\sum_{\tau \in \tau(t_i)} P_i((a'_i, \mathbf{a}^\tau_{-i}), t_i) / |\tau(t_i)| \right] - o(1),$$

for every a_i', where the expectation is over the randomization of the no regret algorithm. Therefore, we have that as $T \to \infty$,

$$\begin{aligned}
\mathbb{E}_{\mathbf{a}^1,\ldots,\mathbf{a}^T}\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}\left[P_i(\sigma^{\text{emp}}(\mathbf{t}),t_i)\right] &= \mathbb{E}_{\mathbf{a}^1,\ldots,\mathbf{a}^T}\left[\sum_{\tau\in\tau(t_i)} P_i(\mathbf{a}^\tau,t_i)/|\tau(t_i)|\right]+\alpha(T)\\
&\geq \mathbb{E}_{\mathbf{a}^1,\ldots,\mathbf{a}^T}\left[\sum_{\tau\in\tau(t_i)} P_i((a_i',\mathbf{a}_{-i}^\tau)t_i)/|\tau(t_i)|\right]+\alpha(T)-o(1)\\
&= \mathbb{E}_{\mathbf{a}^1,\ldots,\mathbf{a}^T}\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}\left[P_i((a_i',\sigma^{\text{emp}}(\mathbf{t})_{-i}),t_i)\right]+\alpha(T)-\beta(T)-o(1)\\
&= \mathbb{E}_{\mathbf{a}^1,\ldots,\mathbf{a}^T}\mathbb{E}_{\mathbf{t}_{-i}\sim\mathcal{D}|_{t_i}}\left[P_i((a_i',\sigma^{\text{emp}}(\mathbf{t})_{-i}),t_i)\right]-o(1),
\end{aligned}$$

which implies that as $T \to \infty$, the iterative no-regret dynamics converge to a B-CCE.

C STACKELBERG POMDP FOR μSPM

In this section, we show how the Stackelberg POMDP can be used to support finding an optimal μSPM. For the purpose of μSPM a game state s contains.

- tf: a flag that indicates whether we're running the policy to compute a new equilibrium, or to get reward.
- σ: the messaging strategies of the agents.
- $\mathbf{v}$: the valuations of agents in the current sub-episode.
- μ: the messages sampled by the current agents' strategies.
- $\mathbf{x}^{t-1}$: the partial allocation in the current sub-episode
- ρ^{t-1}: the agents and items left in the current sub-episode
- ti: the current agent for which we compute the equilibrium (used only when $tf = true$).
- μ_i: the current action taken for computing utility (used only when $tf = true$).
- u: the utility vector of player ti (used only when $tf = true$).

The state transitions are defined as follows:

1. σ initialized.
2. For $t \in [T_1]$ (no-regret sub-episodes):
   - $\mathbf{v}$ is sampled according to the value distribution.
   - μ is sampled according to σ and $\mathbf{v}$.
   - $u_i = \mathbf{0}$ for every agent i.
   - For $i \in [n]$ (agent), $k \in |\mathcal{T}_i|$ (action):
     - $s_0^{t,i,k} = (tf = 1, \sigma, \mathbf{v}, \mu, \mathbf{x} = \emptyset, \rho = (\mathbf{1}_{n+m}), ti = i, \mu_i = k, u = u_i)$.
     - given $s_\ell^{t,i,k}$, the mechanism takes action $a_\ell = (i_\ell, p_\ell)$, where i_ℓ is the next selected bidder and p_ℓ is the vector of posted prices. Bidder i_ℓ chooses a set of items x_ℓ at prices p_ℓ, which is observed by the mechanism ($o_\ell = x_{\ell-1}$). The state $s_{\ell+1}^{t,i,k}$ is obtained by adding the bundle x_ℓ selected by agent i_ℓ to the partial allocation $\mathbf{x}^{\ell-1}$ to form a new partial allocation $\mathbf{x}^\ell$, and the items and agent are removed from the residual setting $\rho^{\ell-1}$ to form ρ^ℓ.
     - After all items are taken or all agents have been visited, we add the utility of agent i to $u_i[k]$
   - Update strategies σ according to the u_is.
3. $s_0^r = (tf = 0, \sigma, \mathbf{v}, \mu, \mathbf{x} = \emptyset, \rho = (\mathbf{1}_{n+m}))$ (u, ti, μ_i are also in the state, but disregarded). $\mathbf{v}$ is sampled from the value distribution, and μ is sampled according to σ and $\mathbf{v}$. This is the initial state of the reward sub-episode.

4. At s^r_ℓ, the mechanism takes action $a_\ell = (i_\ell, p_\ell)$, where i_ℓ is the next selected bidder and p_ℓ is the vector of posted prices. Bidder i_ℓ chooses a set of items x_ℓ at prices p_ℓ, which is observed by the mechanism ($o_\ell = x_{\ell-1}$). The state $s^r_{\ell+1}$ is obtained by adding the bundle x_ℓ selected by agent i_ℓ to the partial allocation $\mathbf{x}^{\ell-1}$ to form a new partial allocation $\mathbf{x}^\ell$, and the items and agent are removed from the residual setting $\rho^{\ell-1}$ to form ρ^ℓ
5. The reward $r(s, a)$ is zero in all states except for terminal states of the reward sub-phase, where no agents or items are left. It can capture any objective of the designer; i.e., social welfare or revenue.

D ITERATIVE MULTIPLICATIVE WEIGHTS

We now describe the multiplicative-weights algorithm that we use to update followers' strategies (Algorithm 2). The algorithm has three parameters: The payoff function of the agents (in our case, this is the payoff function of the game induced a leader's policy ℓ) which determines the game played by our agents, a real number $\varepsilon > 0$ controlling the magnitude of the weight updates, and an integer $T > 1$ determining the number of iterations. Each follower i is assigned a weight matrix w_i of size $|\mathcal{T}_i|$ by $|\mathcal{F}_i|$ where all entries are initialized to 1. The following procedure is repeated T times: First, a type profile $(t_1, .., t_n)$ is sampled from distribution $\mathcal{D}$. Then, for each follower i an action a_i is sampled according to weights $w_i[t_i][a_i]$. Finally, we compute each agent i's payoffs when choosing any action a'_i and when the other agents are playing a_{-i}; we scale each weight $w_i[t_i][a'_i]$ accordingly.

Algorithm 2 No Regret Multiplicative Weights

parameters: The payoff function P, update parameter ε, number of iterations T for each agent i, initialize weights $w_i[t_i][a_i] = 1$ for each type $t_i \in \mathcal{T}_i$ and action $a_i \in \mathcal{A}_i$ **repeat** T **times**
  sample valuation profile $\mathbf{t} = (t_1, .., t_n)$ from distribution $\mathcal{D}$ generate action profile $\mathbf{a} = (a_1, .., a_n)$ where each a_i is sampled with probability $w_i[t_i][a_i] / \sum_{a'_i} w_i[t_i][a'_i]$ **foreach** *agent* $i \in [n]$ **do**
    foreach *action* $a'_i \in \mathcal{A}_i$ **do**
      let P_i be agent i's payoff under payoff function P_i when she has type t_i and the action profile is (a'_i, a_{-i})
      update weight $w_i[t_i][a'_i] = w_i[t_i][a'_i] * (1 + \varepsilon)^{P_i}$
    end
  end
end

E ADDITIONAL NORMAL FORM GAMES

This appendix integrates our experimental section by also considering the first game described in Zhang et al. (2019), which we present in Table 4. Here, the optimal leader action is row C. Indeed, if the leader chooses C, the follower responds with column C and both follower and leader realize their maximum payoff. As we can see from Figure 5, the leader learns to play row C immediately, both when trained with MAPPO and PPO.

F MATRIX DESIGN GAME: CRITIC NETWORK TRAINING PERFORMANCE

In this section, we investigate the training performance of the critic network in our Matrix Design Game experiments (Section 5.2). To do so, we consider how the *value function loss* (i.e., the mean loss of the value function described by the critic network) evolves during training. The results are reported in Figure 6. As we can see from Figure 6, using a centralized critic dramatically improves the accuracy of the critic network, with value loss dropping below 0.2 since the early training stages.

	A	B	C
A	$(15, 15)$	$(10, 10)$	$(0, 0)$
B	$(10, 10)$	$(10, 10)$	$(0, 0)$
C	$(0, 0)$	$(0, 0)$	$(30, 30)$

Table 4: Matrix game 1.

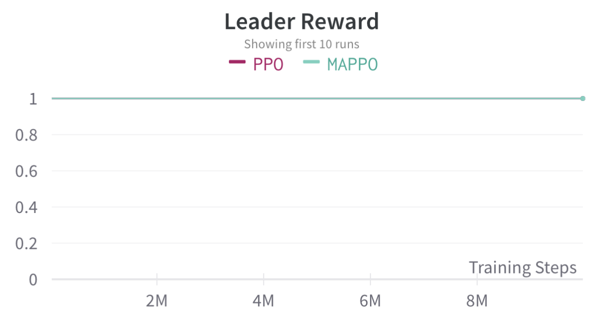


Figure 5: Training curves for matrix game in Table 4 averaged over 10 runs. Rewards are normalized between 0 and 1.

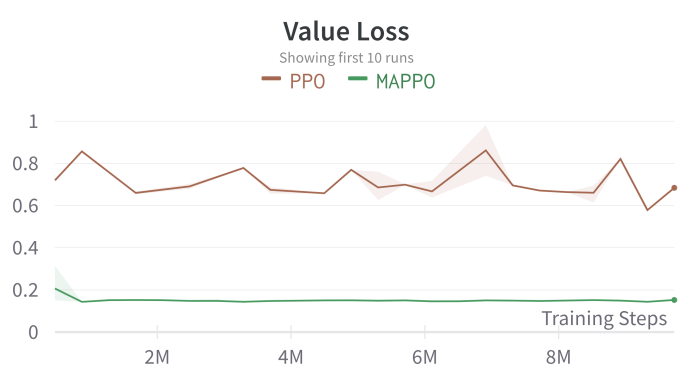


Figure 6: Value loss in Matrix Design Experiments.