ORACLES & FOLLOWERS: STACKELBERG EQUILIBRIA IN DEEP MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Stackelberg equilibria arise naturally in a range of popular learning problems, such as in security games or indirect mechanism design, and have received increasing attention in the reinforcement learning literature. We present a general framework for implementing Stackelberg equilibria search as a multi-agent RL problem, allowing a wide range of algorithmic design choices. We discuss how previous approaches can be seen as specific instantiations of this framework. As a key insight, we note that the design space allows for approaches not previously seen in the literature, for instance by leveraging multitask and meta-RL techniques for follower convergence. We propose one such approach using contextual policies and evaluate it experimentally on standard benchmark domains. Finally, we illustrate the effect of adopting designs outside the borders of our framework in controlled experiments.

1 INTRODUCTION

Stackelberg equilibria are an important concept in economics, and in recent years have received increasing attention in computer science and specifically in the multiagent learning community. In these equilibria, we have an asymmetric setting: A leader who commits to a strategy, and one or more followers who respond. The leader aims to maximize their reward, knowing that followers in turn will best-respond to the leader's choice of strategy. These equilibria appear in a wide range of settings. In security games, a defender wishes to choose an optimal strategy considering attackers will adapt to it (An et al., 2017; Sinha et al., 2018). In mechanism design, we aim to design a mechanism that allocates resources in an efficient manner, knowing that participants may strategize (Nisan & Ronen, 1999; Swamy, 2007; Brero et al., 2021a). More broadly, many multi-agent system design problems can be viewed as Stackelberg equilibrium problems: we as the designer take on the role of the Stackelberg leader, wishing to design a system that is robust to agent behavior.

We are particularly interested in Stackelberg equilibria in sequential decision making settings, i.e. stochastic Markov games, and using multi-agent reinforcement learning techniques to learn these equilibria. In this paper we:

1. Introduce a new theoretical framework for framing Stackelberg equilibria as a multi-agent reinforcement learning problem,
2. Discuss how a range of existing approaches fit into this paradigm, as well as where there remain large unexplored areas in the design space,
3. Reveal a novel approach to accelerating follower best-response convergence, borrowing ideas from multitask and meta-RL, including an experimental evaluation, and
4. Give two important conditions for Stackelberg convergence, and demonstrate experimentally how things can fail when these conditions are not met.

Our main theorem (Theorem 1) allows a black-box reduction from learning Stackelberg equilibria into separate leader and follower learning problems. This framing encompasses and generalizes several prior approaches from the literature and gives a large design space beyond what has been explored previously. Our second main technical contribution is applying contextual policies, a common tool in multitask and meta-RL, to the follower learning problem (Wang et al., 2016). In doing so, followers can generalize, and quickly adapt to leader policies. We validate this approach in

experiments and show performance in line with previous results, with greatly reduced sample complexity compared to previous inner loop-outer loop approaches. We also show that violating either of two conditions of our theorem, the *leader invariance* property and a *Markov property*, can lead to complete failure of the learning process, and that this failure is consistent for different underlying algorithms.

In the remainder of the paper, we will introduce Stackelberg equilibria and Markov games in Section 2. In Section 3, we motivate and define our framework for learning Stackelberg equilibria using multi-agent RL and discuss its scope and limitations. We define our novel contextual policy oracle in 3.2. Finally, in Section 4, we empirically evaluate our proposed new method, as well as show how the violation of some necessary conditions leads to learning failure.

1.1 Prior Work

Learning Stackelberg Equilibria. Most prior work on Stackelberg equilibria focus on single-shot settings such as normal-form games, a significantly simpler setting that Markov games, often in security games. A broad line of work focuses on *computing* Stackelberg equilibria, such as Paruchuri et al. (2008); Xu et al. (2014); Blum et al. (2014); Li et al. (2022). Among the first works on *learning* Stackelberg equilibria was Letchford et al. (2009), who focus on Bayesian games. Peng et al. (2019) give results for matrix games with sample access only. Wang et al. (2022) show an approach differentiating through the so-called KKT conditions, again for normal-form games. Bai et al. (2021) give lower and upper bounds on learning Stackelberg equilibria in general-sum games, including "bandit RL" games with one step for the leader, and sequential decision-making for the followers. Few works in this area consider Markov games: Zhong et al. (2021) show algorithms that find Stackelberg equilibria in Markov games, but assume myopic followers, a significant limitation compared to the general case.

Mechanism Design. One of the first works specifically discussing Stackelberg equilibria in a learning context is Swamy (2007), who design interventions in traffic patterns. More recently, several strands of work have focused on using multi-agent RL techniques to learn optimal mechanism design, often framing this is a bi-level or inner-outer-loop optimization problem. Zheng et al. (2022) use a bi-level RL approach to design optimal tax policies in a simulated world but without provably Stackelberg properties. Yang et al. (2022) use a meta-gradient approach in a specific incentive design setting. Balaguer et al. (2022) use an inner-loop outer-loop gradient descent approach for mechanism design on iterated matrix games (which we also use as our main experimental testbed). They mainly focus on the case where both the environment transition as well as the follower learning behavior is differentiable, and otherwise fall back to evolutionary strategies for the leader. Brero et al. (2021a;b; 2022) use an inner-outer loop approach, using regret minimization for followers and policy gradient methods for a leader agent, which they call the *Stackelberg POMDP*, primarily aimed at indirect mechanism design. Interestingly, of these only the Stackelberg POMDP line of work explicitly mentions Stackelberg equilibria. As a direct corollary of our own work, we show that neither Balaguer et al. (2022) nor Zheng et al. (2022) give Stackelberg equilibria, but Balaguer et al. (2022) could do so with minor modifications.

2 Preliminaries

Markov games. We consider partially observable stochastic Markov games, essentially a multi-agent generalization of a partially observable Markov Decision Process (POMDP).

Definition 1 (Markov Game)**.** *A Markov Game $\mathcal{M}$ with n agents is a tuple $(S, A, T, r, \Omega, O, \gamma)$, consisting of a state space S, an action space $A = (A_1, ..., A_n)$, a (stochastic) transition function $T : S \times A \to S$, a (stochastic) reward function $r : S \times A \to \mathbb{R}^n$, a set of observations $\Omega = (\Omega_1, ..., \Omega_n)$, a (stochastic) observation function $O : S \times A \to \Omega$, and a discount factor γ.*

At each step t of the game, every agent i chooses an action $a_{i,t}$ from their action space A_i, the game state evolves according to the joint action $(a_{1,t}, \ldots, a_{n,t})$ and the transition function T, and agents receive observations and reward according to O and R. An agent's behavior in the game is characterized by its policy $\pi_i : o_i \mapsto a_i$, which maps observations to actions. (We focus here mainly on this memory-less case.) Each agent in a Markov Game individually seeks to maximize

its own (discounted) total reward $\sum_t \gamma^t r_i(s_t, a_{i,t}, a_{-i,t})$. This gives rise to the usual definitions of Nash equilibria (NE), correlated equilibria (CE), and coarse correlated equilibria (CCE), which we do not repeat in full here. If there is additionally a prior over types for each agent (which we will take to influence the agent's reward function only), this leads to analogous definitions for Bayesian equilibria (B-NE, B-CE, B-CCE).

Note that strategies in Markov games and in each of these equilibrium definitions are policies: A pair of policies π_1, π_2 in a two-player Markov game is a Nash equilibrium if neither agent can increase their expected total reward by unilaterally deviating.

Stackelberg Equilibria. Unlike all the above equilibrium concepts, a Stackelberg equilibrium is not symmetric: There is a special player, the leader, who commits to their strategy first; the other player (the follower) then chooses their best strategy given the leader's choice of strategies. This makes the leader potentially more powerful.

Example 1. *In a game often called the "battle of the sexes," you and I wish to have dinner together, but you prefer restaurant A (deriving happiness* 2*, but I only get happiness* 1*), and I prefer restaurant B (I get happiness* 2*, you get happiness* 1*)—but we would both rather eat together at our less-preferred venue, than to eat separately (we both get happiness* 0*). Table 1 shows the payoff matrix of this game. There are two pure Nash equilibria in this game: We both go to restaurant A, or we both go to restaurant B. But there is only a single Stackelberg equilibrium (per leader): If you commit to going to restaurant A, then my only best response is to also go to restaurant A. In doing so I receive happiness* 1*, whereas my only alternative would be to eat alone at restaurant B for happiness* 0*. Notice that this hinges on the leader strictly committing to their choice of restaurant.*

Table 1: "Battle of the Sexes" game.

2,1	0,0
0,0	1,2

This Stackelberg concept again also extends to Markov games: Here a leader agent L decides on their strategy, and the remaining (follower) agents observe and best-respond to this strategy. The leader seeks to maximize their own reward, considering that followers will always best-respond to their choice of strategy. For instance, in an iterated prisoner's dilemma (Robinson & Goforth, 2005), a leader might commit to a tit-for-tat strategy, in turn leading the follower to cooperate.

We are interested in multi-follower settings, where Stackelberg equilibria have also been defined Nakamura (2015); Zhang et al. (2016); Liu (1998); Solis et al. (2016); Sinha et al. (2014); Wang et al. (2022); Brero et al. (2021a). With multiple followers, the "best response" of the followers can be an ambiguous term. Any choice of leader strategy s_L induces a Markov game between F_{s_L} between the followers, which could feature multiple equilibria and equilibria of different types, such as Nash, correlated and coarse correlated equilibria, each giving rise to a corresponding Stackelberg-Nash, Stackelberg-CE and Stackelberg-CCE concept. We handle this in our definition through an oracle (see also e.g. Wang et al. (2022): For any choice of leader strategy, we assume we are given a follower equilibrium (or a probability distribution over equilibria), $\mathcal{E}(\mathcal{F}_{s_L})$, by an oracle $\mathcal{E}$.

Definition 2 (Stackelberg equilibrium)**.** *Given a Markov Game $\mathcal{M}$ and a follower best-response oracle $\mathcal{E}$, a leader strategy s_L together with a tuple of follower strategies $\mathbf{s}_F$ is a* Stackelberg equilibrium*, if s_L maximizes the leaders expected reward under the condition that follower strategies are drawn from $\mathcal{E}(\mathcal{F}_{s_L})$:*

$$s_L \in \arg\max_{s_L} \mathop{\mathbb{E}}_{\mathbf{s}_F \sim \mathcal{E}(\mathcal{F}_{s_L})} \Big[\sum_t \mathbb{E}\big[r_L\big(s_t, a_L, \mathbf{a}_F\big)\big] \Big], \tag{1}$$

where the second expectation is drawing actions and state transitions from their respective policies $s_L, \mathbf{s}_F$ and transition function T, and the reward function is r, all as in Definition 1. If the follower oracle $\mathcal{E}$ gives a Nash equilibrium in the induced game F_{s_L}, we call this a Stackelberg-Nash equilibrium, and similar for CE and CCE.

This use of an oracle is convenient not only for the definition, but also a useful abstraction for a single follower. What we have in mind for practical purposes is an algorithm that computes or learns the follower best-response equilibrium, $\mathcal{E}(\mathcal{F}_{s_L})$, given the leader strategy s_L, often using RL. In the remainder of the paper, when we say "oracle" this is what we mean.

3 A General Framework for Stackelberg Equilibria in Multi-Agent RL

Several approaches have been proposed to learning Stackelberg equilibria in Markov games, or to use multi-agent RL for mechanism design in such settings. A main aim of our work is to elucidate commonalities between these approaches, and to delineate what is required to guarantee Stackelberg equilibria. For instance, most of the existing approaches use (reinforcement or no-regret) learning to implement the follower best-response, effectively arriving at an *"inner-loop-outer-loop system"*: The leader performs one update to their policy, then the followers perform many updates to theirs until they converge to a best response, then this repeats. Is this the only possible approach? Can you mix-and-match leader and follower approaches at will? One approach for leader learning is reinforcement learning, where the gradient of the leader policy is estimated from sampled trajectories (Brero et al., 2021a)—this is in contrast to global approaches such as direct differentiation of the leader policy in a world where everything is differentiable or evolutionary strategies (Balaguer et al., 2022), which modify the leader policy as a whole based on total episode reward, without looking at what happens at each step. Some RL approaches (Brero et al., 2021a; 2022) incorporate the followers' learning dynamics into the leader's view of the environment. Is it necessary that the leader can see this adaption process? Or could we also have follower best-respond to the leader immediately on the first step it takes? Some approaches for mechanism design do not explicitly mention Stackelberg equilibria, but seem very similar to approaches that do (Zheng et al., 2022; Balaguer et al., 2022); do those approaches converge to a Stackelberg equilibrium? In this section we develop a common framework that answers these questions, and provide a common language to categorize the various strands of research in this area.

A key idea is that we can separate the problem into a leader's learning problem, and an implementation of the follower oracle. Our framework informally, provides two insights. One, for *any* follower oracle and *any* construction of the leader problem, a solution is a Stackelberg equilibrium as long as the reward given to the leader is that from the original Markov game with best-responding followers. This encompasses a wide range of leader learning approaches, including direct gradient descent, optimization, and evolutionary strategies. At the same time it gives very simple conditions to ensure Stackelberg solutions, and delineates approaches that do and do not give these. Two, for any query-based oracle implementation, we can construct the leader problem as a POMDP, amenable to solving through RL methods. Since most follower oracle implementations are query-based (or can be approximated through queries), this again encompasses a wide range of possible approaches. Together, these simple conditions open a wide design space, which includes existing approaches but also provides a much wider field of possibilities, one of which we demonstrate in this paper.

We now make this formal. The general case of the following theorem is intentionally kept close to the definition of a Stackelberg equilibrium, and we will discuss implications of this after the theorem statement.

Theorem 1. *Given a Markov Game $\mathcal{M}$ and a follower equilibrium oracle $\mathcal{E}$, then:*

[General Case] *If*

1. *the leader locally aims to solve a single-agent optimization problem $\mathcal{L}(s_L)$, and*
2. *for each choice of leader policy s_L, $\mathcal{L}$ computes the follower best-response $\mathcal{E}(s_L)$, and*
3. *$\mathcal{L}(s_L)$ evaluates the leader policy s_L against the follower best-response $\mathcal{E}(s_L)$ in $\mathcal{M}$, i.e. the value of $\mathcal{L}(s_L)$ is $r_L(s_L, \mathcal{E}(s_L))$ in $\mathcal{M}$,*

then an optimal solution s_L^* to $\mathcal{L}$ together with the follower best-response $\mathcal{E}(s_L^*)$ form a Stackelberg equilibrium in $\mathcal{M}$.

[Query-Oracle Special Case] *Furthermore, if additionally*

4. *the follower oracle implementation $\mathcal{E}$ only requires query access to s_L, i.e. values $s_L(o)$ for one or more observations $o \in \mathcal{O}_L$ from $\mathcal{M}$, and*
5. *leader policies s_L are invariant, i.e. acting the same during queries as they would in $\mathcal{M}$,*

then $\mathcal{L}$ can be constructed as a POMDP.

Proof. See Appendix A. We briefly note for the following discussion that the construction in the query-oracle case constructs episodes in $\mathcal{L}$ by using an *initial segment* in which the oracle queries s_L, and a *final segment* in which an episode of $\mathcal{M}$ is played by s_L and the $\mathcal{E}(s_L)$ computed in the initial segment. This is formalized in the proof in the Appendix. □

Theorem 1 unifies several approaches from the literature. The main theorem of Brero et al. (2021a) can be seen as a special case of the query-oracle case. However, even the query-oracle special case of our theorem significantly generalizes that result to allow for *any* query-based follower oracle. Furthermore, the general case of the theorem extends this to approaches that do not require the leader problem to be a POMDP. While the general case follows very easily from Definition 2, it still provides a very powerful insight. For instance, the "good shepherd" approach (Balaguer et al., 2022) performs gradient descent directly on the leader policy, either by differentiating through the environment and followers, or by estimating the gradient using evolutionary strategies. Neither requires the problem to be a POMDP. At the same time, the general case of Theorem 1 applies, and it tells us that the "good shepherd" approach in its current form may not give Stackelberg equilibria: it accumulates leader reward even during follower learning, i.e., when followers are not (yet) playing $\mathcal{E}(s_L)$ (see Footnote 1). Further, the theorem does not make any assumptions about the type of access to the environment and follower oracle, and can thereby encompass approaches that only require sample access as well as those that have access to a description, and approaches where followers or the environment are differentiable.

Leader Invariance. We draw particular attention to the requirement that the leader policy is invariant, i.e., acting the same during the initial segment (i.e. oracle queries) and the final segment (i.e. original game) of the constructed leader POMDP $\mathcal{L}$. This has not been stated explicitly in previous works, but is a critical part of ensuring convergence to the correct equilibrium. If the leader policy were to act differently during the oracle queries, it could "trick" followers into suboptimal behavior that gives the leader better reward but is not a Stackelberg equilibrium. For instance, in an iterated prisoners dilemma, a leader could pretend to be playing tit-for-tat during oracle queries, leading followers to cooperate; and could then defect during the actual game. We show this experimentally. Invariance is easily ensured if the leader policy cannot distinguish queries and real play, which is generally true for memory-less policies. Alternatively the leader policy can be explicitly constrained to be invariant, e.g. through an appropriate loss term.

Memory. We state the query-oracle case of Theorem 1 for memory-less leader policies, i.e. leader policies that map directly from observations to actions. This is without loss of generality because for leader policies that use memory we may take the view that the leader policy operates on belief states, mapping belief state to action. In this view, the theorem applies as-is, and we query the leader policy on belief states. This would work well, for instance, if leader memory was implemented through a sufficient statistic. Alternatively, if we want to treat memory as intrinsic to the leader policy, queries become sequences of observations. In this view, the proof applies *mutatis mutandis*. The main technicality in this case is to reset internal state of the leader policy between queries, so that queries are well-defined. This is also important in order to ensure the leader invariance conditions (an unrestricted LSTM could easily allow a leader to distinguish queries from real game).

3.1 Solving the Leader Problem

Theorem 1 does not assume a specific approach to solve the leader problem $\mathcal{L}$; rather, it gives us a broad array of tools to solve Stackelberg problems in practice. One particular set of tools lie in the query-oracle special case: by constructing $\mathcal{L}$ as a POMDP, the entire suite of existing (deep) RL algorithms become available. We later show experimental results using PG, PPO and DQN, but this is by no means exhaustive. A reasonable question to ask is whether the POMDP property is actually required for RL algorithms to solve these problems. For instance, if we implemented the follower oracle $\mathcal{E}$ in a way that is not part of the leader problem $\mathcal{L}$, essentially skipping the "initial segment" used in the construction of $\mathcal{L}$ in the special case of Theorem 1, would RL still be able to solve the leader problem? Experimentally, we will show that this can indeed fail.

3.2 IMPLEMENTING THE FOLLOWER ORACLE

A key insight is that Theorem 1 works for *any* implementation of a follower oracle. Some of the prior literature focuses entirely on the leader problem and assumes a literal oracle for follower best-response, sometimes also requiring this be differentiable. Practical follower oracle implementations in the literature can broadly be grouped into two categories.

Optimization-based approaches, which have access to a description of the environment (e.g. a payoff matrix), have been used in simple one-shot settings such as (non-iterated) matrix games, where a description of the leader policy and environment can be used to compute an optimal follower response. This is often coupled with optimization for the leader policy, for instance in Paruchuri et al. (2008); Xu et al. (2014).

Learning-based approaches, which require only sample access to the environment, and potentially the leader policy. This approach is taken in some of the single-shot Stackelberg literature and most of the RL-based Stackelberg literature: every time the leader policy is updated to a new policy, $s_{L,\text{new}}$, followers use a standard learning algorithm, for instance RL or regret minimization in the Markov game case, to learn in the induced game $\mathcal{F}_{s_{L,\text{new}}}$. Any convergence guarantee for the follower algorithm (e.g., CCE in no-regret, or Nash in V-Learning in some conditions) translates to the equivalent Stackelberg-CCE or Stackelberg-Nash equilibrium. Follower weights can be re-initialized randomly after each leader update, which makes the environment stationary, or trained without this (possibly requiring less training, in the case the leader policy only changed a little). It is worth noting that our framework using an RL oracle "looks like" leader and follower are simultaneously learning to act in a typical multi-agent RL system, just at different timescales However we find it useful to think of this setup in terms of leader learning as separate from follower oracle, and there are important differences from standard multi-agent RL: The leader should receive reward only when followers are playing their best-response equilibrium, and the entire follower learning process should be one episode for the leader.

Going beyond this, Theorem 1 suggests a wide design space for implementing the follower oracle. As a key contribution, we explore using multi-task and meta-RL as a means of implementing the follower oracle. This is both to illustrate the power of Theorem 1 as a way to think about Stackelberg learning, as well as due to the advantages of the approach over existing ones.

Multi-Task and Meta-RL: Contextual Policies. We can recognize that the follower games, $\mathcal{F}_{s_L}$, are in fact a family of related problems. For this reason, the follower oracle problem can be seen as a multitask or meta-RL problem, and solved using techniques from those fields. We make use of *contextual policies* (Wang et al., 2016), where a context ω describes the task an agent is supposed to solve. In our case, the context provides the specific MDP among a family of MDPs a follower finds itself in, and ω is a description of the leader policy. This context, ω, is concatenated to the follower agent's observation $o_{i,t}$, and agent i observes $(o_{i,t}, \omega)$ at timestep t.

We focus on settings with a state space small enough to allow the leader policy to be fully understood with a small number of queries (i.e., the leader's action for every possible state), and we directly use the leader's response to a fixed set of queries as the context ω. For instance, in an iterated prisoner's dilemma, we ask the leader three questions: "How do you act on the initial step of the game?", "How do you act if the opponent cooperated in the previous step?" and "How do you act if the opponent defected in the previous step?" Clearly, if these are the only three possible states, this is sufficient to characterize the leader policy. We further use a two-stage training approach. In Phase 1, we train the follower against a different, randomized leader policy in each episode. By the end of this phase, the policy is able to best-respond to all possible leader policies. In Phase 2, we train a leader policy against this follower, where the leader is queried at the beginning of each episode. In the iterated matrix game setting in our experiments, we explicitly define the context, ω, in this way. For settings where this is not possible, the multitask and meta-RL literature provide a range of approaches that infer context, often using recurrent networks (Wang et al., 2016; Mishra et al., 2017; Duan et al., 2016; Rakelly et al., 2019; Zintgraf et al., 2019; Humplik et al., 2019).

3.3 RELATION TO EXISTING APPROACHES IN THE LITERATURE

Previous approaches for Markov games have largely focused on no-regret and policy gradient learning to implement follower oracles, coupled with either RL or direct gradient descent methods for

Table 2: Situating different approaches within the framework of Theorem 1. Approaches marked * do not fully satisfy the conditions of the theorem. Approaches marked in bold indicate approaches that apply to general Markov games, i.e., that are sequential for both leader and followers.

	Leader Learning Approach		
Oracle Implementation	Optimization, Search	Direct Gradient Descent, Evolutionary Strategies	RL
N/A - Oracle Assumed Given	Letchford et al. (2009) Peng et al. (2019)	Wang et al. (2022)	Zhong et al. (2021)
No-Regret			**Brero et al. (2021a)**
RL	Bai et al. (2021)	**Balaguer et al. (2022)*** **Yang et al. (2022)***	**Zheng et al. (2022)***
Multitask / Meta-RL			**new**

the leader. Brero et al. (2021a) use no-regret dynamics to implement the follower oracle inside the leader's episode rollout. They then use standard RL techniques to solve the resulting leader POMDP. Balaguer et al. (2022) use gradient methods to implement the follower oracle. In the case where both followers and world dynamics are differentiable they directly differentiate the leader policy (as opposed to estimating its gradient using sampled trajectories). For the non-differentiable case they use evolutionary strategies (Salimans et al., 2017). Interestingly, they seem to accumulate leader reward throughout the entire learning phase of the follower. This puts their approach outside the scope of our Theorem 1, and may give the leader the wrong optimization target.[1] Our understanding is that if the leader were to optimize for its final reward at the end of follower learning instead, their approach would fall within Theorem 1 and yield a Stackelberg equilibria.Zheng et al. (2022) similarly use two-level RL design, using policy gradient to learn each of the follower oracle and the leader policy. While they do not specifically mention Stackelberg equilibria, this would be the target equilibrium condition in their taxation-policy design setting. Interestingly, they use a curriculum learning approach that can be seen as a rudimentary form of the contextual policy meta-learning approach that we develop in this paper. As with Balaguer et al. (2022), Zheng et al. (2022) also seem to accumulate leader reward, putting them outside the assumptions of Theorem 1. It is also not clear if their implementation of leader memory could violate the leader invariance requirement. However, it may be possible to adapt their approach to fall within the theorem by adjusting leader reward and resetting leader memory between follower queries / learning iterations.

4 EXPERIMENTS

We evaluate our contextual policy approach in this section, and further experimentally show the boundaries of Theorem 1. Our main positive finding is that the contextual policy approach together with standard RL for the leader can match the performance of state-of-the-art approaches, but with greatly reduced sample complexity compared with (Balaguer et al., 2022). We further demonstrate that violating the conditions of Theorem 1 can lead to learning failure. We now describe our test environments and the algorithm used. Appendix B gives further details and full hyperparameters.

Environments. We evaluate our contextual policy approach and general framework on an ensemble of iterated symmetric matrix games, such as the iterated prisoner's dilemma (Robinson & Goforth, 2005). We choose these games as an evaluation domain as they present a significant step up in complexity from previous approaches that give explicit Stackelberg guarantees, in that both leader and followers face a sequential decision-making problem. Previous works that explicitly give Stackelberg equilibria either used settings in which followers only had a single decision to make at the beginning of each episode of $\mathcal{M}$ (Brero et al., 2021a), or had myopic, i.e. horizon-1, followers (Zhong et al., 2021), or had leaders only make a single decision per episode (Bai et al., 2021); all

[1]Imagine for instance a setting where the follower has a quickly-learnable strategy that gives medium reward to the leader, and a slow-to-learn strategy that gives high leader reward. Clearly the latter would be the Stackelberg equilibrium, but the former could give higher reward averaged throughout the learning phase.

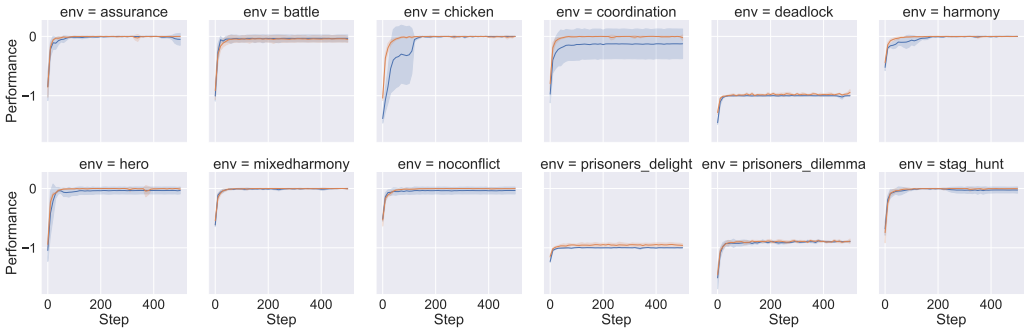

Figure 1: Performance of contextual-policy approach on 12 canonical symmetric iterated matrix games for PG (blue) and PPO (orange) leader algorithms for five random seeds.

significantly simplifying the problem. We mainly compare to the approach of Balaguer et al. (2022) and the baselines therein, because they use these same canonical games for evaluation, allowing for like-for-like comparison. In these environments, we play a matrix games for $n = 10$ steps per episode, and give agents a one-step memory. Therefore, there are five states in these Markov games: one for the initial steps of each episode, and four for later steps depending on the two agents' previous actions. At each step, each agent has a choice of two actions (e.g. "cooperate" or "defect").

Algorithm. We focus on the contextual policy meta-learning approach described in subsection 3.2 for followers, and standard RL for the leader: At the beginning of each episode, the leader is queried (as part of the episode rollout) for its action in each possible state of the environment. Its responses are then concatenated to the follower observation. In a pre-training phase, we train the follower against randomly sampled leader policies. In the main training phase, we then train the leader against the follower meta-policy. Algorithm 1 in the Appendix details this in pseudo-code. An advantage of the generality of our framework is that it is agnostic to which specific RL algorithm is used. We generally use a standard policy gradient (PG) algorithm for the followers, although our results do not depend on this specific choice.

4.1 Results

Performance. Figure 1 shows the performance of our contextual policy approach. We show results using PG and PPO for the leader, and plot per-step rewards realigned to match the reporting of Balaguer et al. (2022) to alllow direct comparison. We see that performance largely matches that of their "good shepherd" approach, and by extension also matches or outperforms all their baselines (c.f. Figures 1 and 2 therein). PG in some games shows some variance between random seeds, whereas PPO learning is very stable across seeds. Both approaches achieve better than tit-for-tat performance on prisoner's dilemma. We further note that the meta-RL approach is highly sample efficient. We use around 5000 episodes for pretraining, and the leader training generally converges after around 1000 episodes, for a total of 60k environment steps sampled. In comparison, Balaguer et al. (2022) indicate that they use 500k follower steps.

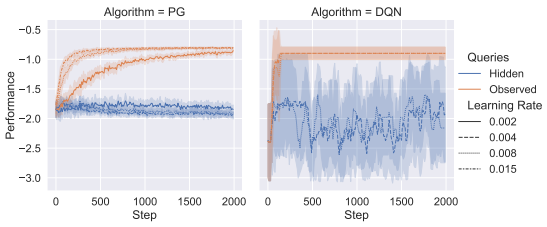

Figure 2: Performance of contextual-policy approach with hidden and rolled-out oracle queries in iterated prisoners dilemma.

Non-POMDP Fails. An interesting question we asked earlier is whether rolling out the follower queries into the leader episode to form a POMDP is strictly necessary. Experimentally we can answer this in the affirmative. Figure 2 shows the performance of our approach on iterated prisoner's dilemma. We show our standard setting where queries are part of the leader episode, as well as a setting where they are hidden from the leader. The hidden-queries setting fails to learn a sensible behavior. This is consistent across learning rates, and across algorithms (PG and DQN pictured).

Non-Invariant Leader Fails. We also experimentally illustrate the leader invariance condition in Theorem 1 in the iterated prisoner's dilemma setting. For simplicity, we emulate memory for the leader policy by concatenating a binary variable to its observation, set to 0 during the first five steps of each episode (the queries), and 1 afterwards.[2] As can be seen in Figure 3, when given access to this additional variable, the leader gains significantly higher reward. The leader policy effectively learns to act as if it was playing tit-for-tat during the queries, thereby inducing the follower to respond by cooperating; the leader then always defects during the actual game, thereby achieving maximum reward. This is not a Stackelberg equilibrium.

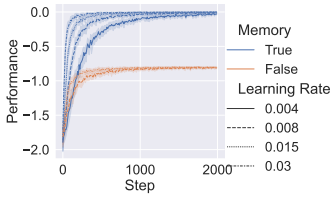

Figure 3: Leader reward with an invariant and a non-invariant policy in iterated prisoners dilemma.

5 Conclusion and Discussion

Conclusion: We have introduced a general framework for using multi-agent RL approaches to find Stackelberg equilibria in Markov games, and discussed how this encompasses several approaches in the literature while also conveying a much larger design space. As suggested by Theorem 1, we have proposed and evaluated a novel approach to Stackelberg learning that uses contextual policies to implement the follower oracle, showing similar performance to previous approaches at much reduced sample complexity in experiments. Finally we showed in controlled experiments that violating the leader invariance and Markov conditions of Theorem 1 can lead to failure of these approaches.

Discussion: In addition to the technical results, we would like to offer a more high-level interpretation of the framework. A useful way to think about learning Stackelberg equilibria in Markov games is that they are, in a way, two problems in one: One, how does my strategy, i.e., choice of policy, affect the best-response of other agents? Two, how does my interactions with the environment, i.e., actions at each step, affect the reward I (and others) get? These are two very different problems, even operating at different levels—entire policy, versus action at each step. In a way, Theorem 1 is giving ways to reconcile the best-response "meta-level" and the environment-interaction "RL problem." In the general case, using techniques such as direct gradient descent or evolutionary policies, we focus on the best-response meta-level and either ignore the environment interaction (in evolutionary strategies) or subsume them inside an end-to-end differentiation (in direct policy gradient). In contrast, in the query-oracle special case, we focus on the environment interaction RL problem, and implicitly work the follower best-response into this. One way of looking at the contextual-policy follower oracle is that it makes the latter more feasible, by greatly reducing the number of leader queries compare to real environment interaction.

We hope that this contextual policy work will enable Stackelberg RL approaches to scale up to richer settings, both through the explicit, contextual-policy approach taken in this paper, as well as approaches that infer context through recurrent networks. Beyond this, we hope the framework of Theorem 1 will inspire novel ways of thinking about Stackelberg RL. One potential avenue for future work that we are excited about is to study approaches that explicitly take into account both the "meta-level" and "environment-interaction" problems outlined above. We believe that doing so could enable Stackelberg RL to scale to much more complex scenarios and open novel applications to it. If successful, this may enable automated learning of system design beyond traditional security games and mechanism design.

[2] A neural network could learn to extract the same discriminator from a step counter, and a recurrent network could easily learn to keep such a counter.

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. Hyperparameters used are also listed in the appendix. We plan to release the source code of our experiments under an open-source license upon acceptance.

REFERENCES

Bo An, Milind Tambe, and Arunesh Sinha. Stackelberg security games (SSG) basics and application overview. *Improving Homeland Security Decisions*, pp. 485, 2017.

Yu Bai, Chi Jin, Huan Wang, and Caiming Xiong. Sample-efficient learning of Stackelberg equilibria in general-sum games. *Advances in Neural Information Processing Systems*, 34:25799–25811, 2021.

Jan Balaguer, Raphael Koster, Christopher Summerfield, and Andrea Tacchetti. The good shepherd: An oracle agent for mechanism design. *arXiv preprint arXiv:2202.10135*, 2022.

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

Gianluca Brero, Darshan Chakrabarti, Alon Eden, Matthias Gerstgrasser, Vincent Li, and David C. Parkes. Learning Stackelberg equilibria in sequential price mechanisms. In *Proc. ICML Workshop for Reinforcement Learning Theory*, 2021a.

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

Gianluca Brero, Nicolas Lepore, Eric Mibuari, and David C Parkes. Learning to mitigate ai collusion on economic platforms. *arXiv preprint arXiv:2202.07106*, 2022.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl˜2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

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.

Zun Li, Feiran Jia, Aditya Mate, Shahin Jabbari, Mithun Chakraborty, Milind Tambe, and Yevgeniy Vorobeychik. Solving structured hierarchical games using differential backward induction. In *Uncertainty in Artificial Intelligence*, pp. 1107–1117. PMLR, 2022.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Baoding Liu. Stackelberg-Nash equilibrium for multilevel programming with multiple followers using genetic algorithms. *Computers & Mathematics with Applications*, 36(7):79–89, 1998.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Tomoya Nakamura. One-leader and multiple-follower Stackelberg games with private information. *Economics Letters*, 127:27–30, 2015.

Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 129–140, 1999.

Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pp. 895–902, 2008.

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

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

David Robinson and David Goforth. *The topology of the 2x2 games: a new periodic table*, volume 3. Psychology Press, 2005.

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

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

Ankur Sinha, Pekka Malo, Anton Frantsev, and Kalyanmoy Deb. Finding optimal strategies in a multi-period multi-leader–follower Stackelberg game using an evolutionary algorithm. *Computers & Operations Research*, 41:374–385, 2014.

Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018.

Cesar U Solis, Julio B Clempner, and Alexander S Poznyak. Modeling multileader–follower noncooperative Stackelberg games. *Cybernetics and Systems*, 47(8):650–673, 2016.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Chaitanya Swamy. The effectiveness of Stackelberg strategies and tolls for network congestion games. In *SODA*, pp. 1133–1142. Citeseer, 2007.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Kai Wang, Lily Xu, Andrew Perrault, Michael K Reiter, and Milind Tambe. Coordinating followers to reach better equilibria: End-to-end gradient descent for stackelberg games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5219–5227, 2022.

Haifeng Xu, Fei Fang, Albert Xin Jiang, Vincent Conitzer, Shaddin Dughmi, and Milind Tambe. Computing minimax strategy for discretized spatio-temporal zero-sum security games. In *International Joint Workshop on Optimization in Multi-Agent Systems and Distributed Constraint Reasoning (OPTMASDCR) at AAMAS*. Citeseer, 2014.

Jiachen Yang, Ethan Wang, Rakshit Trivedi, Tuo Zhao, and Hongyuan Zha. Adaptive incentive design with multi-agent meta-gradient reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1436–1445, 2022.

Huaqing Zhang, Yong Xiao, Lin X Cai, Dusit Niyato, Lingyang Song, and Zhu Han. A multi-leader multi-follower Stackelberg game for resource management in LTE unlicensed. *IEEE Transactions on Wireless Communications*, 16(1):348–361, 2016.

Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C Parkes, and Richard Socher. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances*, 8(18):eabk2607, 2022.

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.

Luisa Zintgraf, Maximilian Igl, Kyriacos Shiarlis, Anuj Mahajan, Katja Hofmann, and Shimon Whiteson. Variational task embeddings for fast adapta-tion in deep reinforcement learning. In *International Conference on Learning Representations Workshop (ICLRW)*, 2019.

A PROOF OF THEOREM 1

We include here the proof of Theorem 1.

Theorem 1. *Given a Markov Game $\mathcal{M}$ and a follower equilibrium oracle $\mathcal{E}$, then:*

***[General Case]** If*

1. *the leader locally aims to solve a single-agent optimization problem $\mathcal{L}(s_L)$, and*
2. *for each choice of leader policy s_L, $\mathcal{L}$ computes the follower best-response $\mathcal{E}(s_L)$, and*
3. *$\mathcal{L}(s_L)$ evaluates the leader policy s_L against the follower best-response $\mathcal{E}(s_L)$ in $\mathcal{M}$, i.e. the value of $\mathcal{L}(s_L)$ is $r_L(s_L, \mathcal{E}(s_L))$ in $\mathcal{M}$,*

then an optimal solution s_L^* to $\mathcal{L}$ together with the follower best-response $\mathcal{E}(s_L^*)$ form a Stackelberg equilibrium in $\mathcal{M}$.

***[Query-Oracle Special Case]** Furthermore, if additionally*

4. *the follower oracle implementation $\mathcal{E}$ only requires query access to s_L, i.e. values $s_L(o)$ for one or more observations $o \in \mathcal{O}_L$ from $\mathcal{M}$, and*
5. *leader policies s_L are invariant, i.e. acting the same during queries as they would in $\mathcal{M}$,*

then $\mathcal{L}$ can be constructed as a POMDP.

Proof. We show the proof in two parts, first for the general case, then the query-oracle special case.

[General Case] We first show the general case:
Assume s_L^* optimally solves $\mathcal{L}$, i.e. $r_{L,\mathcal{L}}(s_L^*) = \max r_{L,\mathcal{L}}(s_L)$.

By condition 3, the leader's reward in $\mathcal{L}$ is the same as that in $\mathcal{M}$ when the followers play their best-response equilibrium, i.e. $r_{L,\mathcal{L}}(s_L^*) = \max r_{L,\mathcal{L}}(s_L) = \max r_{L,\mathcal{M}}(s_L, \mathcal{E}(s_L))$. This immediately means that s_L^* together with $\mathcal{E}(s_L)$ form a Stackelberg equilibrium in $\mathcal{M}$. Condition 2 is only required implicitly to ensure that followers are playing their best-response equilibrium when the leader strategy s_L is evaluated in $\mathcal{M}$. This shows the general case.

[Query-Oracle Special Case]

Construction of $\mathcal{L}$: Given a Markov Game $\mathcal{M}$ and a follower best-response oracle $\mathcal{E}$ that only requires query access to the leader strategy s_L, define *a new leader POMDP $\mathcal{L}$* as follows:

- **[initial segment]** For an initial number of steps in $\mathcal{L}$, each step performs one query from the follower oracle $\mathcal{E}$: If a given query wishes to determine the leader policy's response to observation o, then the leader will receive o as its observation in $\mathcal{L}$, and the leader's action will be given to $\mathcal{E}$ as the response to its query. The leader will receive no reward in these steps.
- **[final segment]** Once a follower equilibrium $\mathbf{s}_F$ has been determined, the remainder of $\mathcal{L}$ will be constructed from the original Markov game $\mathcal{M}$: We let followers act according to the computed follower equilibrium $\mathbf{s}_F$ and treat them (including all their internal state) as part of the environment.

We now show that $\mathcal{L}$ is a POMDP.

POMDP, setup: Let the state of $\mathcal{L}$ be $z_t = (z_{\mathcal{E},t}, z_{\mathcal{M},t}, z_{F,t})$, the internal state of the follower equilibrium oracle (in the initial part of the $\mathcal{L}$), the state of the original Markov Game, and the internal state of the follower agents (in the final part of the $\mathcal{L}$). In the initial part wlog assume this is $(z_{\mathcal{E}}, 0, 0)$, and in the final part $(0, z_{\mathcal{M},t}, z_{F,t})$.

POMDP, part 1: By assumption, $\mathcal{E}$ only requires query access to s_L, i.e. if at timestep t, the oracle's internal state is $z_{\mathcal{E},t}$ and the oracle issues the query o_t, then the oracle's next internal state $z_{\mathcal{E},t+1}$ is a function of only $z_{\mathcal{E},t}$ and q_t, the leader's response to the query o_t. By the construction of the first

part of $\mathcal{L}$, we have that the leader's observation at timestep t is precisely the oracle query o_t, and so it's action a_t gives the oracle response q_t. Together, we get that the $\mathcal{L}$ state at time $t+1$, z_{t+1}, is a function of only z_t and a_t, showing that $\mathcal{L}$ is a POMDP in the initial part.

POMDP, part 2: In the final part of $\mathcal{L}$, at step t, the Markov Game state is $z_{\mathcal{M},t}$, and leader and follower observations depend only on this state, i.e. $o_{L,t} = o_{L,t}(z_{\mathcal{M},t})$ and $o_{F,t} = o_{F,t}(z_{\mathcal{M},t})$. In turn, both the follower actions $a_{F,t}$ as well as the next follower state $z_{F,t+1}$, only depend on $o_{F,t}$ and the current follower state $z_{F,t}$; therefore both depend only on $z_{\mathcal{M},t}$ and $z_{F,t}$. In turn, the next state of $\mathcal{M}$, $z_{\mathcal{M},t+1}$, depends on leader and follower actions, and therefore only on leader action, $z_{\mathcal{M},t}$ and $z_{F,t}$. Together, it follows that z_{t+1} only depends on z_t and the leader's action $a_{L,t}$, meaning the final part of $\mathcal{L}$ is Markovian.

We have therefore shown that $\mathcal{L}$ as a whole is a POMDP. We now show that an optimal policy in $\mathcal{L}$ forms a Stackelberg equilibrium.

Stackelberg: By the assumption that the leader policy is invariant, we have that if $s_L(o) = a$ in response to an oracle query, then $s_L(o) = a$ in the Markov Game $\mathcal{M}$ as well. Therefore, the follower equilibrium $\mathbf{s}_F$ computed by the oracle at the end of the initial part of $\mathcal{L}$ is a best-response equilibrium to the strategy the leader plays in $\mathcal{M}$ in the final part of $\mathcal{L}$.

Now, by construction of $\mathcal{L}$, the leader reward given any s_L in $\mathcal{L}$ is the same as the leader reward in $\mathcal{M}$ when followers play $\mathbf{s}_F$, and by the above $\mathbf{s}_F$ is indeed a best-response equilibrium, i.e. $r_{L,\mathcal{L}}(s_L) = r_{L,\mathcal{M}}(s_L, \mathbf{s}_F) = r_{L,\mathcal{M}}(s_L, \mathcal{E}(s_L))$. Finally, by optimality of s_L^* in $\mathcal{L}$, $s_L^* \in \operatorname{argmax}(r_{L,\mathcal{L}}(s_L))$, and therefore $s_L^* \in \operatorname{argmax}(r_{L,\mathcal{M}}(s_L, \mathcal{E}(s_L)))$. But this precisely means that s_L^* and $\mathcal{E}(s_L^*)$ form a Stackelberg equilibrium in $\mathcal{M}$. □

B Further Experiment Details

Environments. We use the 12 canonical symmetric matrix games identified in Robinson & Goforth (2005) and also used by Balaguer et al. (2022). During training, we scale rewards to be centered at 0, i.e. taking values -1.5, -0.5, 0.5, 1.5, but we report results offset to match the reward scales used by Balaguer et al. (2022). This has no effect on comparability of results. We use $n = 10$ steps per episode.

Algorithm. Algorithm 1 details the two-phase learning algorithm we use. In all the experiments shown in the main text, we use policy gradient to train the follower meta-policy in the pre-training loop. We use PG (Sutton et al., 1999), PPO (Schulman et al., 2017) and DQN (Mnih et al., 2013; 2015) in the main training loop, as indicated in the respective figures. We use linear models, and disable exploration in the leader policy while pre-training the follower and vice versa. Table B lists the hyperparameters used for each of these algorithms. Any hyperparameters not listed were left at default values in **rllib** version 2.0.0. All experiments were run with a single rollout worker (per experiment), and using Torch.

Implementation and Environment. All experiments were implemented using Ray / RLlib 2.0.0 (Liang et al., 2018). Experiments were run on recent Intel Xeon processors with a single core and 2GB RAM per experiment.

Hyperparameter Tuning. Learning rates and batch sizes were tuned using grid search, with some additional tuning using HyperOpt and BayesOpt Python packages, yielding no further improvement however.

Algorithm 1 Contextual Policy

```
Pre-Training Loop
Initialize follower policy π_F
for each pre-training iteration do
    for each episode per sample batch do
        Sample a random leader policy π_L^r
        for each o_L ∈ O_L do
            Query π_L^r for π_L^r(o_L)
        end for
        Set context ω = π_L^r(o_L), o_L ∈ O_L
        for each episode step do
            Return o_{L,t} to leader, (ω, o_{F,t}) to follower
            Step environment using a_L = π_L^r(o_L), a_F = π_F(ω, o_{F,t})
        end for
    end for
    Update follower policy π_F using collected sample batch using PG/PPO/DQN
end for
Main Training Loop
Initialize leader policy π_L
for each training iteration do
    for each episode per sample batch do
        for each o_L ∈ O_L do
            Query π_L for π_L(o_L)
        end for
        Set context ω = π_L(o_L), o_L ∈ O_L
        for each episode step do
            Return o_{L,t} to leader, (ω, o_{F,t}) to follower
            Step environment using a_L = π_L^r(o_L), a_F = π_F(ω, o_{F,t})
        end for
    end for
    Update leader policy π_L using collected sample batch using PG/PPO/DQN
end for
```

Table 3: Hyper-Parameter Configuration Table

Follower Policy Gradient			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	PG	rollout_fragment_length	100
lr	0.02	train_batch_size	100
iterations	500	batch_mode	complete_episodes
Leader Policy Gradient			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	PG	rollout_fragment_length	100
lr	0.156	train_batch_size	100
iterations	1200	batch_mode	complete_episodes
Leader PPO			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	PPO	rollout_fragment_length	1000
lr	0.008	train_batch_size	1000
entropy_coeff	0.0	sgd_minibatch_size	1000
iterations	500	batch_mode	complete_episodes
Leader DQN			
Hyper-Parameter	**Value**	**Hyper-Parameter**	**Value**
algorithm	SimpleQ	rollout_fragment_length	10
lr	0.001	train_batch_size	1024
learning_starts	5000	exploration_type	ParameterNoise
exploration_initial_stddev	1.0	exploration_random_timesteps	0
iterations	20000	batch_mode	complete_episodes