# Oracles & Followers: Stackelberg Equilibria in Deep Multi-Agent Reinforcement Learning

**Matthias Gerstgrasser**
School of Engineering and Applied Sciences
Harvard University
Computer Science Department
Stanford University
matthias@seas.harvard.edu

**David C. Parkes**
School of Engineering and Applied Sciences
Harvard University
parkes@seas.harvard.edu

## 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 both standard and novel benchmark domains, showing greatly improved sample efficiency compared to previous approaches. Finally, we we demonstrate through theory and experiments the importance of a particular model we adopt for how a follower algorithm is able to interact with the leader's policy.

## 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. They model 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 addition to economic settings such as mechanism design (Nisan & Ronen, 1999; Swamy, 2007; Brero et al., 2021), they play an important role in security games, where a defender wishes to choose an optimal strategy considering attackers will adapt to it (An et al., 2017; Sinha et al., 2018).

In this paper, we are particularly interested in Stackelberg equilibria in sequential decision making settings, i.e., *stochastic Markov games*, and using multi-agent reinforcement learning to learn these equilibria. We would like to highlight two results we believe are of particular relevance to the security community:[1]

1. We introduce a new theoretical framework for framing Stackelberg equilibria as a multi-agent reinforcement learning problem.
2. Inspired by this framework, we introduce a novel approach to accelerating follower best-response convergence, borrowing ideas from multitask and meta-RL, including an experimental evaluation on existing and new benchmark domains.

[1] A longer version this paper focusing on mechanism design problems appeared in ICML 2023.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

These results are of particular relevance to the multi-agent security community, as they enable learning of Stackelberg equilibria to scale up to more complex domains than was previously possible, including in security games. We believe this could enable novel real-world applications of work in security games.

**Prior Work** Our work builds on prior approaches on learning Stackelberg equilibria, including in security games Letchford et al. (2009); Peng et al. (2019); Wang et al. (2022); Bai et al. (2021); Zhong et al. (2021) as well as RL for (indirect) mechanism design Zheng et al. (2022); Yang et al. (2022); Balaguer et al. (2022); Brero et al. (2021, 2022). We utilize contextual policies (Wang et al., 2016; Mishra et al., 2017; Duan et al., 2016; Rakelly et al., 2019; Zintgraf et al., 2019; Humplik et al., 2019) in our own approach.

## 2 Preliminaries

**Stackelberg Equilibria.** Unlike many other equilibrium concepts (e.g. Nash), 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 strategy. 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.*

We are particularly interested in sequential, i.e. Markov games: Here a leader agent $L$ decides on their policy (i.e. strategy), and the remaining (follower) agents—knowing the leader's choice of policy—best-respond. The leader seeks to maximize their own reward, considering that followers will best-respond. For instance, in an Iterated Prisoners' Dilemma (Robinson & Goforth, 2005), a leader might commit to a Tit-for-Tat strategy, in turn leading the follower to cooperate.

Table 1: "Battle of the Sexes" game.

| | |
|---|---|
| 2,1 | 0,0 |
| 0,0 | 1,2 |

Typically, a Stackelberg equilibrium is formally defined using a max-min-style condition: The leader maximizes its own reward knowing that the follower best-responds, i.e., maximizes its own reward, with the leader-follower dynamic giving the order of the two nested max-operators. As a key innovation in this work, we instead use a statement of the follower best-response through an oracle abstraction. An oracle definition has been used before in order to extend Stackelberg equilibria to multiple followers Nakamura (2015); Zhang et al. (2016); Liu (1998); Solis et al. (2016); Sinha et al. (2014); Wang et al. (2022); Brero et al. (2021). In contrast, we use the oracle abstraction to greatly simplify the statement and proof of our main theorem, while simultaneously generalizing it beyond prior approaches. In turn, this allows us to develop a novel Meta-RL approach in the second part of the paper.

This oracle will later be realized as an algorithm.

**Definition 1** (Stackelberg equilibrium)**.** *Given a Markov Game $\mathcal{M}$ and a follower best-response oracle $\mathcal{E}$, a leader strategy $\pi_L$ together with a tuple of follower strategies $\pi_F$ is a* Stackelberg equilibrium*, if and only if $\pi_L$ maximizes the leader's expected reward under the condition that follower strategies are drawn from $\mathcal{E}(\pi_L)$:*

$$\pi_L \in \operatorname{argmax}_{\pi_L} \mathop{\mathbb{E}}_{\pi_F \sim \mathcal{E}(\pi_L)} \Big[ \sum_t \mathbb{E}[r_L(s_t, a_{L,t}, a_{F,t})] \Big],$$

*where the second expectation is drawing actions and state transitions from their respective policies $\pi_L, \pi_F$.*

## 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 contribution of our paper is to provide a unified framework that elucidates commonalities between these approaches and generalizes to broader principles. We state our main theorem here, and provide a more detailed discussion and proof in Appendix A.

**Lemma 1.** *Given a Markov Game $\mathcal{M}$ and a follower equilibrium oracle $\mathcal{E}$, let $\mathcal{L}_{\mathcal{M}}$ be the learning problem the leader faces. If:*

1. *for each choice of leader policy $\pi_L$, $\mathcal{L}$ computes the follower best-response $\mathcal{E}(\pi_L)$, and*
2. *$\mathcal{L}(\pi_L)$ evaluates the leader policy $\pi_L$ against the follower best-response $\mathcal{E}(\pi_L)$ in $\mathcal{M}$, i.e. the value of $\mathcal{L}(\pi_L)$ is $r_L(\pi_L, \mathcal{E}(\pi_L))$ in $\mathcal{M}$,*

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

**Theorem 1.** *Given a Markov Game $\mathcal{M}$, and a follower equilibrium oracle $\mathcal{E}$, if in addition to the conditions of Lemma 1, the follower oracle $\mathcal{E}$ is a query oracle (Definition* **??***), then the leader learning problem $\mathcal{L}$ can be constructed as a POMDP.*

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

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

A key novelty in our framework is that it generalizes to arbitrary approaches for the leader and followers. This allows us to develop our Meta-RL approach in the next section, but also allows us to categorize existing approaches by how they handle leader and follower learning. Table 3 gives an overview of select prior approaches.

## 4 Meta-RL for Stackelberg RL

Going beyond existing approaches, 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. In this approach, we recognize that the follower games, $\mathcal{F}_{\pi_L}$, are in fact a family of related problems. We then make use of *contextual policies* (Wang et al., 2016) to learn a meta-policy for the follower that can best-respond to different leader policies. In a second training stage, we then train a leader policy against this meta-follower, which greatly accelerates the learning process compared to prior nested outer-inner loop approaches. Appendix G provides a more detailed discussion.

### 4.1 Experiments

We evaluate this Meta-RL approach on both a benchmark iterated matrix game domain, comparing to existing approaches, as well as on a novel Atari 2600-based domain that is significantly more

challenging. In the first, our main positive finding is that our approach can match or exceed prior approaches at greatly improved sample efficiency. In the latter, we show for the first time a positive result using a principled Stackelberg approach on a state-of-the-art general RL benchmark domain. In particular, in the matrix game domain our approach converges in generally under 100k timesteps, where prior approaches use millions or billions of timesteps. In the Atari domain, our approach is able to learn exactly two separate equilibria corresponding to either agent being the Stackelberg leader. Figure 1 shows performance and behavior of agents in both eqilibria. Note in particular that the approach learns the optimal prices of 0.25 and 0.75 exactly. Appendix H provides more detail on the experimental domains and results.

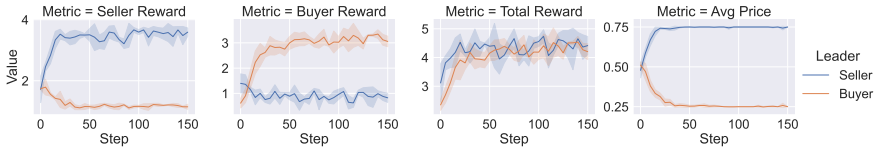


Figure 1: Performance and behavior of PPO+Meta-RL on the Atari 2600 bilateral trade scenario. Plots show results for two distinct Stackelberg equilibria: Agent 1 (seller) as leader (blue curves) and agent 2 (buyer) as leader (orange).

## 5 Discussion

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. We show the necessity of our POMDP construction by showing that RL against followers that immediately best respond can fail, an important and surprising result by itself. We also show through theory and experiments the importance of the query-oracle construction in Theorem 1, showing that without this RL algorithms are provably unable to learn. Concretely, we give a new approach to Stackelberg learning that uses Meta-RL to implement the follower oracle and leverages the framework provided by Theorem 1. Through Meta-RL, we are able to match or exceed the final mean episode reward of previous approaches at a greatly improved speed of convergence. Meta-RL also enables Stackelberg learning in richer domains as demonstrated by application to an Atari 2600-based bilateral trade scenario.

Given the importance of Stackelberg equilibria in security games, we expect Stackelberg learning to enable many new applications to the security field. In particular, the relevance of the results presented here is that they allow Stackelberg learning to be extended to richer settings. This could allow work on Stackelberg Security Games to tackle novel domains that were previously intractable, and could thereby enable new applications of multi-agent security research. Furthermore, in addition to the specific Meta-RL approach we present here, our theoretical framework outlines a wide range of potential future work in algorithmic advances in Stackelberg learning to further improve on the approach we presented.

In closing, we also offer a more high-level interpretation of the theoretical 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. Theorem 1 is giving a way 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 compared to real environment interaction.

## References


An, B., Tambe, M., and Sinha, A. Stackelberg security games (SSG) basics and application overview. *Improving Homeland Security Decisions*, pp. 485, 2017.

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

Balaguer, J., Koster, R., Summerfield, C., and Tacchetti, A. The good shepherd: An oracle agent for mechanism design. *arXiv preprint arXiv:2202.10135*, 2022.

Bergstra, J., Yamins, D., and Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pp. 115–123. PMLR, 2013.

Brero, G., Chakrabarti, D., Eden, A., Gerstgrasser, M., Li, V., and Parkes, D. C. Learning Stackelberg equilibria in sequential price mechanisms. In *Proc. ICML Workshop for Reinforcement Learning Theory*, 2021.

Brero, G., Lepore, N., Mibuari, E., and Parkes, D. C. Learning to mitigate ai collusion on economic platforms. *Advances in Neural Information Processing Systems*, 35, 2022.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. $Rl^2$: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

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

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

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

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

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

Peng, B., Shen, W., Tang, P., and Zuo, S. Learning optimal strategies to commit to. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2149–2156, 2019.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Sinha, A., Fang, F., An, B., Kiekintveld, C., and Tambe, M. Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018.

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

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Wang, K., Xu, L., Perrault, A., Reiter, M. K., and Tambe, M. 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.

Yang, J., Wang, E., Trivedi, R., Zhao, T., and Zha, H. 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.

Zhang, H., Xiao, Y., Cai, L. X., Niyato, D., Song, L., and Han, Z. A multi-leader multi-follower Stackelberg game for resource management in LTE unlicensed. *IEEE Transactions on Wireless Communications*, 16(1):348–361, 2016.

Zheng, S., Trott, A., Srinivasa, S., Parkes, D. C., and Socher, R. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances*, 8(18):eabk2607, 2022.

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

Zintgraf, L., Igl, M., Shiarlis, K., Mahajan, A., Hofmann, K., and Whiteson, S. Variational task embeddings for fast adapta-tion in deep reinforcement learning. In *International Conference on Learning Representations Workshop (ICLRW)*, 2019.

## A Discussion of Main Theorem

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., 2021)—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 leader RL approaches (Brero et al., 2021, 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

the follower best-respond to the leader immediately on the first step it takes? Some approaches for mechanism design do not explicitly mention Stackelberg equilibria (Zheng et al., 2022; Balaguer et al., 2022), but seem very similar to approaches that do; 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 novelty in our framework is that following Definition 1 we separate the problem into a leader learning problem and a follower oracle implementation. Let $\mathcal{L} = \mathcal{L}_{\mathcal{M}}$ be the learning problem that the leader faces. In standard multi-agent RL, $\mathcal{L}$ would simply be the leader's local view of the multi-agent system, considered as a single-agent learning problem (i.e., taking the other agents as being part of the environment). We will now show how to construct $\mathcal{L}$ so as to guarantee a Stackelberg equilibrium. As a warm-up, we will derive two basic conditions on $\mathcal{L}$ that guarantee that an optimal policy in $\mathcal{L}$ forms a Stackelberg equilibrium.

**Lemma 1.** *Given a Markov Game $\mathcal{M}$ and a follower equilibrium oracle $\mathcal{E}$, let $\mathcal{L}_{\mathcal{M}}$ be the learning problem the leader faces. If:*

1. *for each choice of leader policy $\pi_L$, $\mathcal{L}$ computes the follower best-response $\mathcal{E}(\pi_L)$, and*
2. *$\mathcal{L}(\pi_L)$ evaluates the leader policy $\pi_L$ against the follower best-response $\mathcal{E}(\pi_L)$ in $\mathcal{M}$, i.e. the value of $\mathcal{L}(\pi_L)$ is $r_L(\pi_L, \mathcal{E}(\pi_L))$ in $\mathcal{M}$,*

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

This follows in a straightforward way from Definition 1 (see Appendix B for a formal proof). It is also easy to see that these are essentially necessary conditions (modulo transformations and reward shaping, see Appendix F). While Lemma 1 is simple, it already allows us to answer some of the questions we posed initially. In particular, do previous not-explicitly-Stackelberg approaches from the literature give Stackelberg equilibria? They are not guaranteed to, because many of them either do not train followers until convergence (e.g. (Zheng et al., 2022), violating condition 1), or give reward even while followers are still learning (e.g. (Balaguer et al., 2022), violating condition 2).

However, Lemma 1 only applies if we already have an optimal solution to $\mathcal{L}$, and it does not say anything about whether a particular leader learning algorithm, such as RL, will converge on $\mathcal{L}$. In particular, we asked earlier if it is necessary for the leader to be able to see the followers' adaption process, or if we could also have followers best-respond immediately? We first show that "letting the leader see" the followers' adaption process (formally, the operation of the oracle $\mathcal{E}$) makes $\mathcal{L}$ a POMDP, meaning that RL algorithms for the leader can be expected to converge under standard assumptions. Through the oracle abstraction, we show this for a much broader class of follower best-response algorithms than typical learning approaches. Second, we show this is also a necessary condition, i.e., leader RL can fail if followers best-respond immediately and without a visible adaption process. In the following theorem, we show that for *any* query oracle, a suitable construction of the leader problem $\mathcal{L}$ will be a POMDP, i.e. exhibit Markovian state transitions. This POMDP can, in turn, be solved via leader RL to yield a Stackelberg equilibrium, which makes the theorem applicable to settings where it is desirable to use RL to solve the leader learning problem.

**Theorem 1.** *Given a Markov Game $\mathcal{M}$, and a follower equilibrium oracle $\mathcal{E}$, if in addition to the conditions of Lemma 1, the follower oracle $\mathcal{E}$ is a query oracle (Definition* **??***), then the leader learning problem $\mathcal{L}$ can be constructed as a POMDP.*

In the following, we provide the main idea behind this positive result, which relies on the construction of a suitable, single-agent POMDP to model the leader's problem. We defer the full proof of Theorem 1 to Appendix B, where we also give concrete examples.

Given $\mathcal{M}$ and a query oracle $\mathcal{E}$ we define *a new leader POMDP* $\mathcal{L}$ as follows: The action and observation space for the leader in $\mathcal{L}$ are the same as those in $\mathcal{M}$. $\mathcal{L}$ is then constructed in two parts. First, the leader is queried as often as is required to compute a follower equilibrium by the oracle $\mathcal{E}$; then an episode from the original Markov game $\mathcal{M}$ plays out:

- **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 $\pi_F$ 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 $\pi_F$ and treat them (including all their internal state) as part of the environment.

If the follower oracle is implemented using RL, i.e., both leader and followers use RL, then the initial segment is simply one or more episodes of $\mathcal{M}$ where the followers are learning, and the final segment is one episode from $\mathcal{M}$ where the followers have converged. This formalizes, in a general way, the "train followers until convergence for every leader policy update" approach seen in prior work (Brero et al., 2022). The innovation is that we generalize to *any* query-based algorithm implementing the follower oracle, not just followers using learning algorithms, and this will be crucial in the next section. (And note that we further generalize even to non-query follower algorithms in Lemma 1, which applies to the case of non-RL leaders; see Appendix D.1).

The POMDP property of the construction in Theorem 1 gives us strong evidence that that outer-loop inner-loop RL approaches have a solid theoretical foundation. However, such approaches also have drawbacks (such as very long and sparse-reward episodes for the leader). The remaining question is whether this visibility is necessary, or if we could skip the initial segment in the construction in Theorem 1 and have followers best-respond immediately on the first step the leader takes. For the first time, we can answer this question clearly: RL against immediately-best-responding opponents can provably diverge.

**Theorem 2.** *There exists a Stochastic Markov Game, $\mathcal{M}$, where neither tabular Q-learning nor policy gradient can learn the optimal policy for the leader when the follower agent immediately best-responds.*

We prove Theorem 2 in Appendix E, and also confirm the effect experimentally in Appendix D.1 in the context of the Iterated Prisoners' Dilemma. This result is highly surprising, as training against a best-responding follower at first may seem like a natural approach that one might consider using. In the proof of Theorem 2, we see that the RL algorithm fails due to missing counterfactuals. In particular, the key idea is that the best response to a leader's policy may depend on its behavior on all possible states, including states not visited when the follower actually best-responds. For instance, in iterated prisoner's dilemma, the threat of retaliation to a defection is crucial. But if a follower best-responds by always cooperating, the defection state will never be visited, and will not lead to corresponding updates in the RL algorithm. We use this insight to then construct an instance where all other possible visits of the defection state (from a non-optimal leader policy, or random exploration) will lead to RL updates that "point away" from the optimal policy.

Collectively, Lemma 1, Theorem 1, and Theorem 2 allow us to answer the last remaining question we posed initially: Can we mix-and-match arbitrary approaches for the leader and followers? The answer is "Yes, we can." The only restriction is that if we use RL for the leader (or another approach that hinges on a Markov property), then we need to use a query-based approach for the followers and the initial-final segment construction from Theorem 1.[2]

Appendix C further details how to extend Theorem 1 to leader policies with memory, and details a crucial condition of *leader invariance*. This invariance condition requires that the leader policy act the same during oracle queries as it does during real play. In the memory-less case this follows immediately in theory, but is an important implementation detail in practice. As an illustration, we show in Appendix D.4 an example where a seemingly innocuous step counter being made part of the leader's observation leads to learning failure.

## B Proof and Discussion of Theorem 1

We include here the proof of Lemma 1 and Theorem 1.

[2]Note that this is strictly a special case, as there are also other approaches on either side that fall outside this. For instance, one could use evolutionary strategies for the leader, coupled with a non-query-based optimization approach for the followers. See for instance Appendix D.1 for a relevant experiment.

**Lemma 1.** *Given a Markov Game $\mathcal{M}$ and a follower equilibrium oracle $\mathcal{E}$, let $\mathcal{L}_{\mathcal{M}}$ be the learning problem the leader faces. If:*

1. *for each choice of leader policy $\pi_L$, $\mathcal{L}$ computes the follower best-response $\mathcal{E}(\pi_L)$, and*
2. *$\mathcal{L}(\pi_L)$ evaluates the leader policy $\pi_L$ against the follower best-response $\mathcal{E}(\pi_L)$ in $\mathcal{M}$, i.e. the value of $\mathcal{L}(\pi_L)$ is $r_L(\pi_L, \mathcal{E}(\pi_L))$ in $\mathcal{M}$,*

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

*Proof.* Assume $s_L^*$ optimally solves $\mathcal{L}$, i.e. $r_{L,\mathcal{L}}(s_L^*) = \max r_{L,\mathcal{L}}(\pi_L)$. By condition 2, 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. $r_{L,\mathcal{L}}(s_L^*) = \max r_{L,\mathcal{L}}(\pi_L) = \max r_{L,\mathcal{M}}(\pi_L, \mathcal{E}(\pi_L))$. This immediately means that $s_L^*$ together with $\mathcal{E}(\pi_L)$ form a Stackelberg equilibrium in $\mathcal{M}$. Condition 1 is only required implicitly to ensure that followers are playing their best-response equilibrium when the leader strategy $\pi_L$ is evaluated in $\mathcal{M}$. This shows the general case. □

We now show the main theorem.

**Theorem 1.** *Given a Markov Game $\mathcal{M}$, and a follower equilibrium oracle $\mathcal{E}$, if in addition to the conditions of Lemma 1, the follower oracle $\mathcal{E}$ is a query oracle (Definition* **??***), then the leader learning problem $\mathcal{L}$ can be constructed as a POMDP.*

*Proof.* Given a Markov Game $\mathcal{M}$ and a follower best-response oracle $\mathcal{E}$ that only requires query access to the leader strategy $\pi_L$, define *a new leader POMDP $\mathcal{L}$* as follows: The action and observation space for the leader in $\mathcal{L}$ are the same as those in $\mathcal{M}$. $\mathcal{L}$ is then constructed in two parts. First, the leader is queried by the oracle $\mathcal{R}$; then an episode from the original Markov game $\mathcal{M}$ plays out:

- **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 $\pi_F$ 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 $\pi_F$ 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 $\pi_L$, 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 $\pi_L(o) = a$ in response to an oracle query, then $\pi_L(o) = a$ in the Markov Game $\mathcal{M}$ as well. Therefore, the follower equilibrium $\pi_F$ 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 $\pi_L$ in $\mathcal{L}$ is the same as the leader reward in $\mathcal{M}$ when followers play $\pi_F$, and by the above $\pi_F$ is indeed a best-response equilibrium, i.e. $r_{L,\mathcal{L}}(\pi_L) = r_{L,\mathcal{M}}(\pi_L, \pi_F) = r_{L,\mathcal{M}}(\pi_L, \mathcal{E}(\pi_L))$. Finally, by optimality of $\pi_L^*$ in $\mathcal{L}$, $\pi_L^* \in \text{argmax}(r_{L,\mathcal{L}}(\pi_L))$, and therefore $\pi_L^* \in \text{argmax}(r_{L,\mathcal{M}}(\pi_L, \mathcal{E}(\pi_L)))$. But this precisely means that $\pi_L^*$ and $\mathcal{E}(\pi_L^*)$ form a Stackelberg equilibrium in $\mathcal{M}$. □

**Discussion** We intentionally stated the lemma and theorem in a fairly abstract manner, so as to be general and cover a wide range of possible oracle implementations. The theorem may be more readily understood through concrete examples:

In the simplest case, the follower oracle is implemented using **reinforcement learning**, i.e. the leader and follower(s) all use RL. In this scenario, the initial segment of $\mathcal{L}$ is simply one or more episodes of $\mathcal{M}$, where the followers are learning. The final segment is an episode of $\mathcal{M}$ when the followers have converged and are not learning anymore. This "looks" very similar to a standard independent-learning multi-agent RL setup, but with some crucial differences (necessary due to conditions 2 and 3 from the general case of the theorem): For the leader, the initial and final segment form one single episode (but are treated as multiple episodes for the followers), and the leader does not receive reward in the initial-segment episodes. A variant of this is done in Brero et al. (2022), and variations that do not strictly follow conditions 2 and 3 (and thus do not strictly guarantee Stackelberg) are common in the literature as discussed in section 4.

A related case is a follower oracle implemented using a different learning approach, such as **no-regret dynamics** in Brero et al. (2021). In this case, the leader POMDP looks similar to the RL case, except in the initial segment the followers are now learning using a no-regret algorithm such as multiplicative weights. This is qualitatively different, as these algorithms explore in a more systematic way than RL algorithms do.

Note that for the purposes of Theorem 1 and in contrast to typical multi-agent RL setups, in both the above situations we view the leader learning as separate from the follower (reinforcement or no-regret) learning. The latter in our model is an algorithm to implement the follower best-response oracle, and it is useful to think of it in this way. In this view, the initial segment isn't a joint leader-follower multi-agent system, it is the follower oracle algorithm querying the leader policy. In the above cases the follower oracle happens to use the same or a similar algorithm as we use to learn the leader behavior, but a crucial consequence of our statement of Definition 1 and Theorem 1 is that this need not be the case. Indeed, for any follower oracle algorithm that only uses query access to the leader policy, we can use the same construction. For instance, in our **Meta-RL** approach, we use a fixed set of queries $o_0, \ldots, o_k$ to define the context for the meta-follower. In this case, the initial segment in the leader POMDP $\mathcal{L}$ will always be the same sequence of observations $o_0, \ldots, o_k$. The leader's actions in responses to these observations in turn are used to form the context for the meta-follower in the final segment. The final segment is an episode of $\mathcal{M}$ played between the leader and the meta-follower, whose behavior is informed by the context.

Finally, what would a follower oracle implementation look like that falls outside the query-oracle special case of Theorem 1 (but within the general case)? *Firstly*, any of the above cases have a non-POMDP counterpart: We could simply omit the initial segment from the leader's training batch (but in a centralized training regime we could still run forward passes through the leader's neural network to answer the follower oracle queries). Crucially, the followers' behavior changes without any action being taken by the leader. This makes such a construction non-Markovian from the leader's point of view (thought the general case of Theorem 1 still applies). However, this is not the most interesting case. *Secondly*, any follower oracle algorithm that makes use of a *description* of the leader policy, rather than query access, would be incompatible with the POMDP construction. For instance, if the leader policy $\pi_L$ is parametrized by weights $\theta$, it is conceivable that a follower oracle algorithm could compute a best response directly from $\theta$. In such a case, it is not possible to roll out the best-response computation into the leader's experiences as we do through the initial segment of

the special case POMDP construction. Without this initial segment, however, learning can provably fail, as Theorem 2 shows

# C Theorem 1: Leader Memory and Leader Invariance

**Leader 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).

**Leader Invariance.** It may be possible to give the leader policy memory beyond the two cases above, i.e. memory with state that carries through between follower oracle queries and/or to real play. In any such cases, it is necessary that the leader policy be *invariant*, meaning it is 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 best-response, and thus 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 in Appendix D.2. 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 could be explicitly constrained to be invariant, e.g. through an appropriate loss term.

# D Limitations of Lemma 1 and Theorem 1

We now present experimental evidence of the limitations of our main Lemma and Theorem. In particular, we will show that violating any of the conditions of the theorem can lead to learning failure.

## D.1 Non-POMDP

An interesting question we asked earlier is whether rolling out the follower queries into the leader episode to form a POMDP is strictly necessary. Theorem 2 formally shows that this is the case, but we also test this experimentally here. Figure 2 shows the performance of our approach on a slightly modified iterated prisoner's dilemma (see Appendix E for full payoff matrices). 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. Note however that this only applies to RL algorithms. An approach that operates directly on the policy space such as Evolutionary Strategies is still able to learn successfully, as shown on the right hand side of Figure 2.

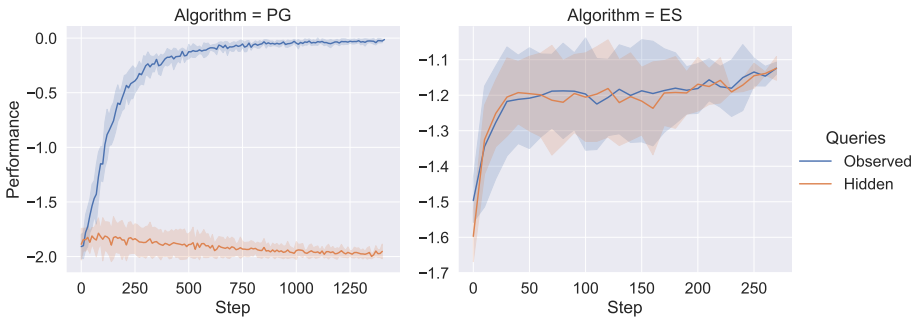


Figure 2: Performance of contextual-policy approach with hidden and rolled-out oracle queries in iterated prisoners dilemma.

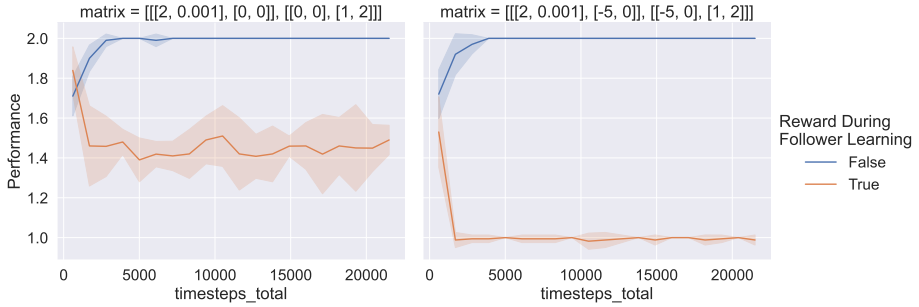


Figure 4: Leader performance with and without reward during follower Q-learning. Clearly the leader fails to learn the sole Stackelberg equilibrium (reward 2.0) if reward is given during follower learning. Plots show reward during actual play only, i.e. without reward during follower Q-learning, as this is the relevant quantity for Stackelberg equilibria.

### D.2 Non-Invariant Leader

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.[3] 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.

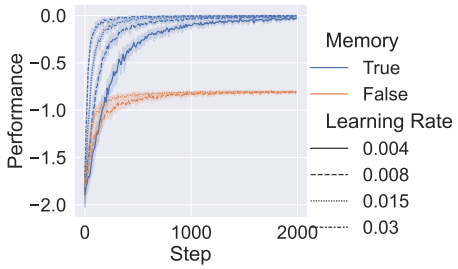


Figure 3: Leader reward with invariant and non-invariant policies in iterated prisoners dilemma.

### D.3 Leader Reward During Follower Learning

One condition of Theorem 1 is that the leader only be evaluated against followers who are best-responding. If the follower oracle is implemented using learning dynamics observable to the leader, this means that the leader must not receive reward during this learning phase. If the leader did receive reward, this could give the leader the wrong optimization target. Imagine for instance a setting where the leader has one strategy choice corresponding to a quickly-learnable follower best-response strategy that gives medium reward to the leader, and another leader strategy choice corresponding to a slow-to-learn follower strategy with high leader reward. We can easily simulate this using a slightly modified version of the "Battle of the Sexes" single-shot matrix game we used as an example in the introduction. In this, we modify the follower reward so that the leader-preferred option gives the follower very little reward. We then couple this with carefully chosen (but entirely reasonable) Q-learning hyperparameters for the follower. As a result, a leader who receives reward during the follower learning phase is not able to reliable learn the correct equilibrium anymore, even in such a simple game, as Figure 4 shows. If we further modify the game to penalize the leader for coordination failure, this can even lead to the leader consistently learning the wrong coordination choice, as the right-hand plot shows.

Notice however that this (reward throughout follower learning) is also a valid target to optimize for, where the leader aims to optimize its expected return taking into account that followers may need some time to adjust to the leader's behavior. In the case of Balaguer et al. (2022) this is the intent, especially with regards to designing mechanisms for human participants as followers.

### D.4 Continuous Follower Learning

Finally, virtually all previous approaches in the literature use some sort of learning dynamics to implement the follower oracle. A tempting way of improving learning speed in such a paradigm would be to retain follower policies between leader updates. That is, if at the end of the leader learning iteration $t$, the follower is best responding using strategy / policy parameters $\phi_{t,\text{end}}$, then instead of initializing follower weights $\phi_{t+1,\text{start}}$ randomly, set $\phi_{t+1,\text{start}} = \phi_{t,\text{end}}$. Under the assumption that the leader policy only changed a little, and the conjecture that therefore the optimal follower policy only changed a little, this should allow follower learning to start from very near the optimum, and thus hopefully require much short inner (follower learning) loops. However, this has some drawbacks. For one, it makes the leader learning problem non-stationary. Beyond this, it can lead to learning failure, if both leader and follower get stuck on a local optimum. Figure 5 shows this in practice on the "Battle of the Sexes" example, where non-resetting follower learning can lead to convergence to the follower-preferred choice rather than the Stackelberg equilibrium.

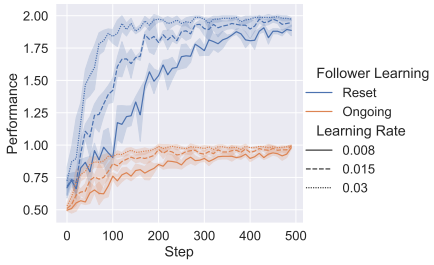


Figure 5: Leader reward with a Q-learning follower on Battle of the Sexes, where the follower initializes a blank Q-table each episode (blue) or keeps their previous Q-table (orange).

## E Proof of Non-POMDP Divergence

We now present a proof of Theorem 2. A priori it is not clear that the query-oracle POMDP construction is strictly necessary, or if standard RL algorithm could also learn without it. Without the POMDP construction, the leader would effectively always play against followers who immediately best respond. The following theorem shows that this cannot work.

[3] 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.

**Theorem 2.** *There exists a Stochastic Markov Game, $\mathcal{M}$, where neither tabular Q-learning nor policy gradient can learn the optimal policy for the leader when the follower agent immediately best-responds.*

*Proof.* We consider a slight variation of the iterated prisoner's dilemma discussed in the main text. Consider payoff matrices $L = \begin{pmatrix} 0 & -2 \\ -1 & -3 \end{pmatrix}$ and $F = \begin{pmatrix} -1 & 0 \\ -3 & -2 \end{pmatrix}$ denoting the payoff to the leader and follower agent respectively. The leader chooses the row, the first row denoting "cooperate" or $C$ and the second row "defect" or $D$, and similarly the follower chooses the column. Notice that these are the standard prisoner's dilemma payoff matrices, except the top and bottom row for the leader have been switched.

Let each agent's observation space be a one-step memory of the *other* agent's previous action, that is, there are three possible observations $o_0$, $o_C$ and $o_D$. At the first step of each episode, both agents observe $o_0$. If at step $t$ the leader cooperates and the follower defects, then at step $t+1$, the leader observes $o_D$ ("other agent defected") and the follower observes $o_C$ ("other agent cooperated"). We also write $s_{CD}$ for this state if we want to refer to both agents. In particular $s_{CD}$ corresponds to leader reward -2 for the leader and 0 for the follower (top right corner of both matrices). This is a simplification of the setting presented in the main text, but without loss of generality in the case of iterated prisoner's dilemma, and also defines a valid stochastic Markov game in its own right. Let us define an episode of our SMG to be $h$ iterations of the matrix game, where $h$ denotes the horizon or episode length of the game. As a preliminary, notice that for any leader policy, the follower best-response is always deterministic. This is easy to check.

It is also easy to see that the optimal leader policy is to cooperate on the first step, and to then play tit-for-tat. That is, if the follower cooperated, the leader cooperates in the following step. If the follower defects, the leader defects in return. If the leader plays this policy, then the follower will in turn always cooperate, leading to leader episode reward 0, clearly the optimum. Using the construction in the query-oracle special case of Theorem 1, this optimal leader policy can be learned using standard RL algorithms. What we will now show is that if the follower best-responds without that construction, i.e. immediately without queries folded into the leader sample batch, then standard RL algorithms will diverge. This is independent of choice of hyperparameters, but as a matter of principle.

Intuitively, the problem is one of a missing counterfactual: Notice that for the leader tit-for-tat Stackelberg equilibrium, it is essential that the leader commits to defecting if the follower defects. But notice also that when the leader plays this tit-for-tat policy, the follower will always best respond, and so the leader will never actually see a follower defection. But this also means that it cannot accumulate a gradient for this hypothetical behavior.

We now make this formal. Consider first the case of (tabular) Q-learning. Let $q(s,a)$ be the (leader's) Q-value of taking action $a$ in state $s$. We let $\alpha$ denote the learning rate and $\gamma$ the discount factor. Given an experience $(s,a,r,s')$ we update Q-values as follows:

$$q(s,a) \quad \leftarrow \quad (1-\alpha)\cdot q(s,a) \quad + \quad \alpha \cdot \big(r + \gamma \max q(s',.)\big) \tag{1}$$

As a convenient shorthand and as a slight abuse of notation, we will define $\theta$ as follows:

$$\theta_s = \begin{cases} 0 & \text{if } q(s,D) \leq q(s,C) \\ 1 & \text{if } q(s,D) > q(s,C) \end{cases} \tag{2}$$

In words, we let $\theta_s = 1$ denote that the current leader policy given the $q(s,a)$ values will defect in state $s$, and 0 if the leader will cooperate in state $s$. We can then write $\theta = (\theta_0, \theta_C, \theta_D)$ for the entire leader policy induced by the current Q-table. $\theta = (0,0,0)$ would denote a leader policy that always cooperates, $\theta = (1,1,1)$ denotes a leader always defecting, and $\theta = (0,0,1)$ denotes the (optimal) tit-for-tat strategy.

Now consider the case of tabular Q-learning with parameter noise exploration. In this, we collect experiences from any of the eight possible deterministic leader policies. Note also that the leader action on the initial step does not affect the follower's best response strategy; and it does not influence Q-table updates for the non-initial observations $o_C, o_D$ (because $o_0$ will never be revisited and so the reward generated from $o_0$ can never appear in a Q-table update or indeed in a reward-to-go calculation

in a policy gradient algorithm). We can therefore disregard the leader's initial action and for brevity focus only on the four cases $\theta = (\star, 0, 0)$, $\theta = (\star, 0, 1)$, $\theta = (\star, 1, 0)$ and $\theta = (\star, 1, 1)$. It is easy to see that for $\theta = (\star, 0, 1)$ the follower best-response is to always cooperate, and for the other three cases it is to always defect. We may therefore encounter experiences of the following form:

$$\begin{aligned}
\theta = (\star, 0, 0) \quad &\rightarrow \quad (o_D, C, -2, o_D) \\
\theta = (\star, 0, 1) \quad &\rightarrow \quad (o_C, C, 0, o_C) \\
\theta = (\star, 1, 0) \quad &\rightarrow \quad (o_D, C, -2, o_D) \\
\theta = (\star, 1, 1) \quad &\rightarrow \quad (o_D, D, -3, o_D)
\end{aligned}$$

It is easy to see that under usual Q-learning update rules and for any choice of learning rate, we will have that in the limit $q(o_D, C) = -2 \cdot g$ (lines 1, 3) and $q(o_D, D) = -3 \cdot g$ (line 4) where $g = \frac{1-\gamma^{h/2}}{1-\gamma}$ is a term from the discount factor $\gamma$. Crucially we have that $q(o_D, C) > q(o_D, D)$, and therefore the policy will converge toward $\theta = (\star, \star, 0)$, which is not optimal. This holds for any choice of learning rate, discount factor and exploration parameters (as any mix of the above trajectories will lead to this).

For the $\epsilon$-greedy case, let $\theta^\epsilon_s = \theta_s + (-1)^{\theta_s}(\epsilon/2)$. That is, if our current Q-table induces the deterministic policy $\theta$, then $\theta^\epsilon_s$ gives the probability of choosing action $D$ in state $s$ in the $\epsilon$-greedy case. It is easy to see that for sufficiently small $\epsilon$ and $\theta^\epsilon_s = (\star, \epsilon, 1-\epsilon)$ the follower best-response is still to always cooperate, and for any other $\theta^\epsilon_s$ the follower best-response is to always defect. Therefore in particular, no matter which way a particular leader action is sampled, the follower will best-respond in the same way (only depending on the leader policy as a whole, not the particular leader action sampled). In turn this means that $q(o_D, C)$ can only continue to accumulate $-2$ terms, and $q(o_D, D)$ can only continue to accumulate $-3$ terms, and the policy will converge toward $\theta = (\star, \star, 0)$, which is not optimal.

To show this for policy gradient, let the leader policy be parametrized by $theta$ as above, i.e. let $\theta_o$ be the probability that the leader policy defects given observation $o$, and $1 - \theta_o$ the probability that the leader cooperates given $o$. Recall the basic REINFORCE gradient update rule:

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \ln \pi_\theta(a_t | o_t) \tag{3}$$

Here $G_t$ denotes the (discounted) "reward to go", i.e. $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots$ as usual. A very similar argument as in the Q-learning case now holds to show that the reward-to-go from cooperating when observing $o_D$ will always be larger in expectation than the reward-to-go from defecting, because $r_t$ when defecting is smaller than $r_t$ when cooperating given $o_D$ and the remainder of the sum in $G_t$ is the same in expectation. This in turn pushes gradients toward cooperation, and away from the optimal tit-for-tat policy. □

The above holds for tabular Q-learning and basic policy-gradient with direct parametrization, but likely can be extended to further RL algorithms such as DQN or actor-critic.

Notice the key difference in the query-oracle POMDP construction: In this, the oracle must query the leader policy for its action given $o_D$ at least once in the initial "oracle" segment of the episode. That action therefore sees as its reward to go the reward from the entire final segment, i.e. the entire episode reward of the original Markov game. Intuitively, the leader gets to see at least one experience where it retaliates on a follower defection and this leading to an entire episode of cooperation and good rewards. Without the oracle query, the leader never gets to see this, and cannot learn from it. It may still see experiences where it retaliates for defection, but these will be from within the actual episode, will not influence follower behavior, and will lead to strictly worse rewards than cooperating. Finally, it is also clear that this only applies to typical RL algorithms that learn on taking actions in individual steps. Approaches that learn on the policy space as a whole, such as evolutionary strategies, are not affected by this (as indeed they never look at individual steps and actions at all).

## F Necessary Conditions for Stackelberg Convergence

It may also be interesting to consider the inverse direction of Theorem 1, i.e. what are necessary conditions that follow from Stackelberg convergence. The resulting theorem is not very strong, but

still informative, as it suggests avenues for future research. Recall that in Theorem 1 we map a Markov game to a single-agent RL problem (POMDP) for the leader. In the general case this is simply taking the leader's view of the original Markov game as-is, and in the query-oracle special case we construct a POMDP that incorporates oracle queries. We then show that a solution to the leader's POMDP together with the follower best-response forms a Stackelberg Equilibrium.

Consider now the reverse: Suppose we are given some mapping from Markov game to leader POMDP, and a guarantee that no matter the original Markov game, an optimal solution to the leader POMDP it maps to forms part of a Stackelberg equilibrium. What needs to be true of any such mapping? We formulate this here in a slightly more general manner, in that we also allow an additional (not necessarily identity) mapping between leader policies in the Markov game and the POMDP.

**Theorem 3** (Necessity)**.** *Suppose we are given mappings $\mathcal{L} : \mathcal{M} \mapsto \mathcal{L}(\mathcal{M})$ and $l : \Pi_{\mathcal{L}} \to \Pi_{\mathcal{M}}$. $\mathcal{L}$ maps any Markov Game $\mathcal{M}$ to a single-agent RL problem, and $l$ maps policies in $\mathcal{L}$ to policies in $\mathcal{M}$. Furthermore suppose that whenever a policy $\pi_{L,\mathcal{L}}$ optimally solves $\mathcal{L}(\mathcal{M})$, then $l(\pi_{L,\mathcal{L}})$ together with $\mathcal{E}(l(\pi_{L,\mathcal{L}}))$ are a Stackelberg equilibrium in $\mathcal{M}$. Then the following two conditions must be true of $\mathcal{L}$ and $l$.*

1. *The leader reward in $\mathcal{L}$ is maximized by the same choice of strategy as the leader reward in $\mathcal{M}$ when followers play $\mathcal{E}(\pi_L)$, i.e.*

$$l\big(r_{L,\mathcal{L}}(\pi_L)\big) \subseteq r_{L,\mathcal{M}}(\pi_L, \mathcal{E}(\pi_L))$$

2. *$\mathcal{L}$ implements a follower equilibrium oracle $\mathcal{E}(\pi_L)$*

*Proof (Sketch).* The first condition immediately follows from the problem statement. For the second condition, consider that given full freedom in choosing $\mathcal{M}$, we can construct $\mathcal{M}$ so as to let $\mathcal{E}$ be any arbitrary function from leader to follower policy space. Similarly, we can choose $\mathcal{M}$ so that $r_L$ is any arbitrary function. Both of these follow from cardinality arguments, and the observation that since Markov games may be partially observable we are essentially unrestricted in the complexity of the Markov game we choose to construct even for small strategy spaces. Since by the first condition $\mathcal{L}$ needs to compute $r \circ \mathcal{E}$, both of which can be arbitrary, it thus also needs to compute $\mathcal{E}$. □

The main difference to the conditions in Theorem 1 is that we can only show that the argmax of the leader reward needs to be that of the original Markov game, not that the rewards need to be identical. This is in a way trivial (of course Theorem 1 still holds if we scaled leader rewards in the leader learning problem by a constant factor), but it also suggests that reward shaping may be a viable technique to accelerate leader learning, potentially still with provable Stackelberg equilibrium guarantees.

## G Additional Details on Meta-RL Approach

In our Meta-RL approach, we recognize that the follower games, $\mathcal{F}_{\pi_L}$, 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 $\omega$ 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 $\omega$ is a description of the leader policy. This context, $\omega$, is concatenated to the follower agent's observation $o_{i,t}$, and agent $i$ observes $(o_{i,t}, \omega)$ at timestep $t$. Crucially, we will construct context $\omega$ through queries of the leader policy, so that we can use the POMDP construction of Theorem 1.

We focus on settings where the leader policy's effect on the follower can be fully understood with a small number of queries, and we directly use the leader's response to a fixed set of queries as the context $\omega$. For instance, in the Iterated Prisoners' 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 a follower meta-policy against a different, randomized leader policy in each episode. By the end of this phase, the meta-policy is able to best-respond to all possible leader policies. In Phase 2, we train a leader policy against

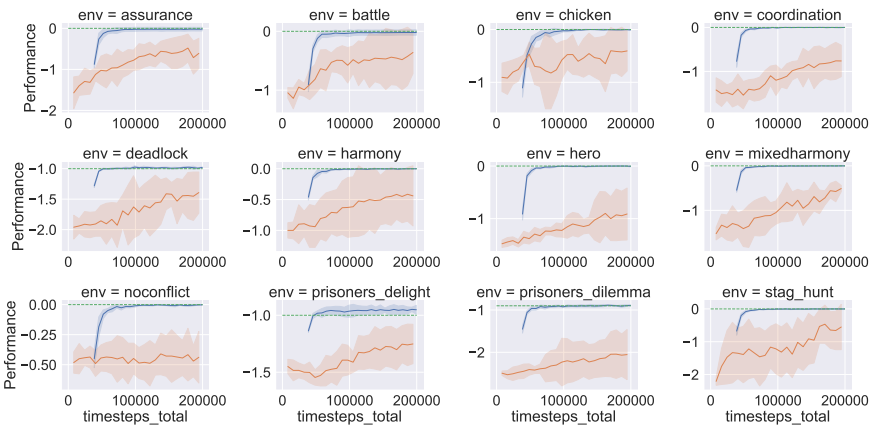


Figure 6: Blue: Mean episode reward of our novel PPO+Meta-RL approach on 12 canonical symmetric iterated matrix games. Orange: PPO+Q-learn Brero et al. (2022). Dashed green: Good Shepherd ES-MD Balaguer et al. (2022) (final, mean episode reward at 1.28B timesteps, estimated from Fig. 2 ibid.)

this follower, where the leader is queried at the beginning of each episode. Note that this two-stage approach is separate from the two-segment construction in Theorem 1. The first training phase is purely a pre-training stage for the follower meta-policy. In effect, it builds a suitable query-based follower oracle. The second training phase trains the leader, and uses the two-segment construction from Theorem 1 in each episode experienced through leader training. Concretely, in Phase 2, every episode looks as follows: First, in the "initial segment" (Theorem 1), the leader sees a fixed sequence of observations $o_0, \ldots, o_k$. The follower does not act in these steps at all. The leader's actions, $a_0, \ldots, a_k$, in response to the sequence of observations, $o_0, \ldots, o_k$, are memorized. Specifically in the iterated matrix game environments we use, this sequence of observations is simply all the possible states of $\mathcal{M}$, and so the leader's response fully characterizes its policy. The actions $a_0, \ldots, a_k$ then form the context $\omega = (a_0, \ldots, a_k)$ for the follower in the "final segment." In this second part of each episode, leader and (meta-) follower act together in an episode of the original game, $\mathcal{M}$. At a given timestep $t$ if the current state is $s_t$, the leader's observation is $o_{L,t} = s_t$, and the (meta-) follower's observation is $o_{F,t} = (s_t, \omega)$. In words, the follower observes its usual observation, but also the context. Through this context, the follower is informed about the leader policy it is playing against, and can best-respond to it.

Because the context $\omega$ was derived through queries that are part of the leader's batch of experiences, this forms a POMDP for the leader by Theorem 1. At the same time, because the follower does not need to learn from scratch how to respond for every leader policy update, we avoid the "inner loop" of typical "outer-inner loop" approaches. For settings where it is not possible to explicitly define the context $\omega$ in this way, the multitask and meta-RL literature provides 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).

## H Experiments

**Environments: Iterated Matrix Games.** We evaluate our contextual policy approach and general framework on an ensemble of iterated symmetric matrix games, such as the Iterated Prisoners' Dilemma (Robinson & Goforth, 2005). We choose these games 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. In these, we play a matrix game for $n = 10$ steps per episode, and give agents a one-step memory. This makes these environments Markov games, with five states: 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"), leading to the next state, e.g. "both cooperated".

Figure 6 shows the performance of our Meta-RL approach using PPO for the leader. We compare against the approaches of Balaguer et al. (2022) and Brero et al. (2022). For our PPO+Meta-RL approach, we plot the combined environment steps used by the meta-follower training plus the leader training on the x-axis. For Balaguer et al. (2022), we estimate performance from Figure 2 therein. Note that this is the eventual performance at the end of training (Balaguer et al. (2022) do not publish learning curves).

In terms of mean episode reward, the final performance largely matches that of the "good shepherd" ES-MD approach, which is expected as both approaches achieve at or very near the theoretical optimum on all games; i.e., they both find the Stackelberg equilibrium of the game. By extension our approach also matches or outperforms all their baselines (c.f. Figures 1 and 2 therein). The more relevant comparison between our approach and "good shepherd" is on speed of convergence, where our Meta-RL approach converges in around 50k environment steps, whereas Balaguer et al. (2022) report performance at 1.28 billion environment steps in the ES-MD case. We give further details on this comparison with Balaguer et al. (2022) in Appendix K.

We also see that our approach outperforms the PPO+Q-learn approach of Brero et al. (2022). In Appendix K we show the PPO+Q-learn approach training for significantly longer, and see that when it does converge it does so around 500k environment steps at the earliest, whereas for most of the harder cases it still has not nearly reached optimal performance at 2M timesteps. We again note that our Meta-RL approach shows greatly improved sample efficiency.

**Environments: Bilateral Trade on Atari 2600.** As a second, significantly higher-dimensional and challenging domain, we present a bilateral trade scenario on a modified Atari 2600 game (a state-of-the-art domain in single-agent RL). We use a two-player version of *Space Invaders*, and introduce an artificial resource constraint: Each agent can only fire in the game if they have a bullet available. Initially, neither player has any bullets available. Throughout the episode, we give bullets to player 1, one at a time at stochastic intervals. Player 1 can then choose to offer the sell this bullet to player 2 by offering them a price, or Player 1 can choose to use the bullet themselves. Player 2 in turn can choose to accept or reject a particular offer at a particular price. If a trade takes place, the sales price is added to player 1's reward, and subtracted from player 2's reward. Additionally, we introduce a reward scale imbalance: Each time player 1 successfully shoots an alien invader, they get a reward of 0.1. However each time player 2 shoots an alien, they get a much higher reward of 1.0. Noting that even well-trained AI agents do not hit every single shot they take, we should still expect that player 2 be able to generate just under 1.0 reward from each bullet they fire, and player 1 a much smaller reward of just under 0.1.

There is more total reward generated if player 1 sells all their bullets to player 2, with the difference the "gains from trade" in economics. However, this is not a mechanism design setting (there is no mechanism), and also that there are two Stackelberg equilibria: If player 1 is the leader, then their optimal strategy is to offer bullets to player 2 at just under player 2's average utility per bullet. Player 2 will best respond by accepting the trade, still generate small positive reward, and player 1 will receive almost the entirety of the gains from trade. In the second Stackelberg equilibrium, player 2 is the leader. Player 2's optimal strategy is to refuse any price higher than just above player 1's average utility per bullet; and player 1's best response is to offer to sell at that (low) price. In this scenario, player 1 will be left with little more reward than had they kept and used the bullets themselves, and player 2 will receive almost all the gains from trade.

Figure 1 shows that the Meta-RL algorithm is able to successfully learn this optimal behavior for both equilibria. In this experiment we use discrete prices $(0, 0.25, 0.5, 0.75.1.0)$ for compatibility with the discrete Atari environment, so the results shown are the exact optimum.

# I Further Experiment Details: Iterated Matrix Games

**Environments** We use the 12 canonical symmetric matrix games identified in Robinson & Goforth (2005) and also used by Balaguer et al. (2022). We construct Markov games from these matrices by concatenating multiple iterations into an episode, and giving both agents one-step memory of both agents' action in the previous step. We use $n = 10$ steps per episode. Table K shows the payoff matrices for all the Markov games, reported on the same scale as the figures. 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.

**Algorithm.** We focus on the contextual policy meta-learning approach described in subsection 4 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.

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 K 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.

**Equilibrium Verification.** At the end of every experiment, we freeze the leader policy and further train the follower policy for $n = 50$ iterations. Unlike in the pre-training phase, we here train them only against the specific leader policy trained in the main training loop. This is to further verify that the policies indeed form a Stackelberg equilibrium, and in particular that the follower meta-policy is best-responding to the trained leader. If this is the case, we should not see any change in leader or follower performance in this post-training phase. If the follower meta-policy was *not* already best-responding to the leader, we may see an increase in follower performance during this post-training phase. In all of the experiments in this paper (except the ones designed to show failure modes) we see no follower improvement, i.e. behavior consistent with a Stackelberg equilibrium. This is not shown in the training curves in the figures, but can be reproduced from the source code.

**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 the HyperOpt Python package (Bergstra et al., 2013), yielding no further improvement however.

## J Further Experiment Details: Atari 2600

**Environment.** We modify the Atari 2600 game "Space Invaders". We read from emulator RAM to detect when a shot has been fired, and by which player. Separately in a Python wrapper we keep a count of how many shots each player has available. We decrement this whenever we detect that the player fired a shot. If the Python variable keeping track of the available bullets reaches zero, we overwrite the player action that is fed to the Atari emulator to not-firing. Both players start with zero available bullets, but we increment the bullets available to player 1 at stochastic intervals for up to a total of five times per episode.

We implement a bilateral trade between agents: The selling agent may offer a price, and the buying agent may choose to accept this price.

**Neural Network Architecture.** This is implemented by augmenting both action and observation space, both providing a dictionary of both the underlying Atari action/observation, as well as the new economic action and observations.

The action space contains the original Atari action, as well as the trading action. For the seller, the trading action is picking one of several discrete price points, where we choice $n = 5$ price points

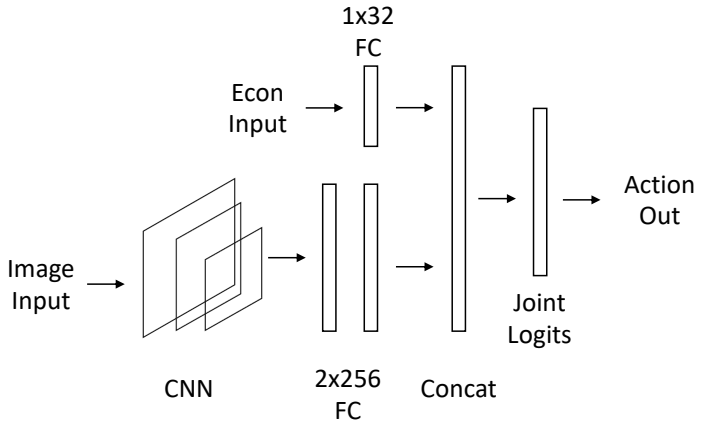


Figure 7: Neural Network Architecture used in the Atari 2600 bilateral trade experiments.

ranging from 0 to 1 in 0.25-step increments. For the buyer, instead of giving a discrete buy / don't-buy action, we let the buyer policy set a maximum price it is willing to buy. If the offered sales price is below the maximum buying price of the buyer, the trade happens, and the price paid is that set by the selling agent. It is easy to see that this is equivalent to letting the buying agent observe the price offer and respond with acceptance or rejection. We chose this implementation as it makes implementing the follower oracle easier when the buyer is Stackelberg leader, but it does not affect the outcome.

In the observation space, we provide a Dictionary to each agents containing both the original Atari 2600 image observation, as well as all the relevant economic information (number of bullets the agent currently has available, if applicable price offered by the other agent, whether a trade is current being proposed). In the neural network, we run these economic features through a separate fully connected layer, which feeds into a joint logits layer. The Atari input is run through default RLlib CNN and fully connected layers. Figure 7 shows this neural network architecture as a diagram.

**Algorithm.** We use standard PPO for both the leader and follower. Hyperparameters were taken from RLlib tuned examples and are listed at the end of Table K. For convenience, we initialize weights of the CNN and default RLlib FC layers to weights obtained from training agents in the unmodified game. This speeds up training, but is not strictly necessary. We utilize the same Meta-RL approach as we do in the iterated matrix game experiments: We first train a meta-follower. In this phase, we let the gameplay actions of the leader agent be controlled by an agent trained on the unmodified game, but we randomize the leader's economic actions. Once this meta-follower training has finished, we train the leader. In this phase, the meta-follower weights are frozen, and only the leader policy is trained. In the Atari experiments, we let the meta-follower query the leader immediately before each trade rather than at the start of the episode, as this allows us to fold the queries into the trading exchange.

## K Further Details on Performance Comparisons

In Figure 6 we compare our Meta-RL approach with the PPO+Q-learn approach of Brero et al. (2022) and the ES-MD approach of Balaguer et al. (2022).

For Brero et al. (2021), we implement follower Q-learning using information therein. Hyperparameters for both the leader and the follower were tuned using the HyperOpt package (Bergstra et al., 2013). In Figure 6 we plot learning curves up to 200k timesteps, as our approach converges before that point. We show in Figure 8 learning curves until 2M timesteps. We can see that in some cases PPO+Q-learn eventually converges to the optimum, while in the majority of cases this still has not happened by 2M timesteps.

For Balaguer et al. (2022), we estimate their performance from Figure 2 therein. Notice that that figure is *not* a learning curve, but represents a single inner loop at the end of their training procedure.

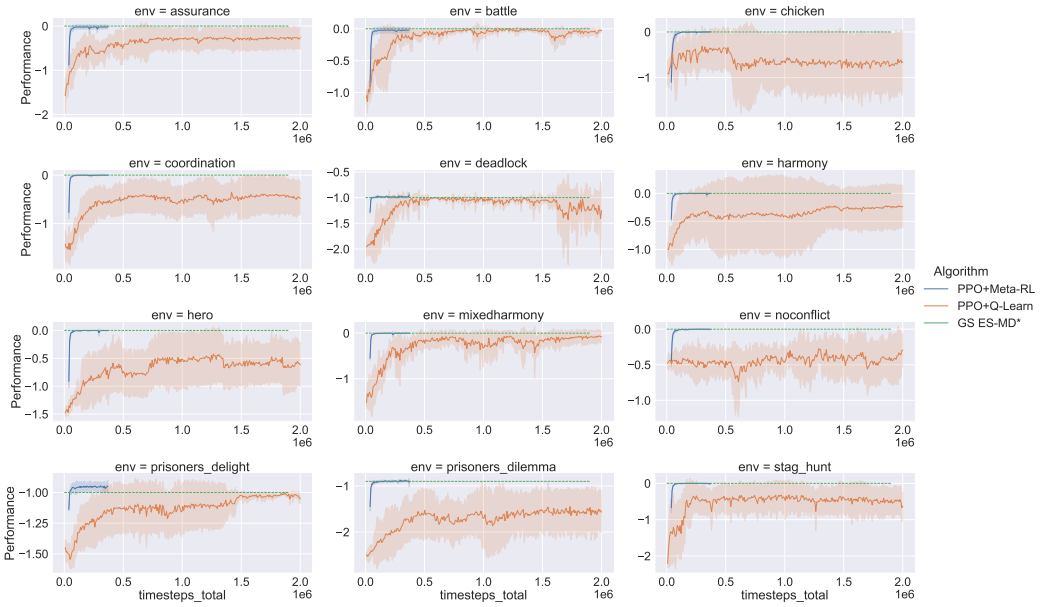


Figure 8: Performance on symmetric matrix games (see Figure 6) up to 2M timesteps.

In the ES-MD case, Balaguer et al. (2022) report their performance after 1.28 billion environment steps. In the Diff-MD case, a comparison of sample complexity is difficult, as that approach uses a description of the environment rather than sample access. The closest we can come to a like-for-like comparison is noting that Balaguer et al. (2022) report performance for Diff-MD after 500k computed expected episode returns with 10-step episodes. In some sense this could be seen to be equivalent to 5M environment steps as a lower bound.

**Algorithm 1** Contextual Policy

**Pre-Training Loop** Initialize follower policy $\pi_F$ each pre-training iteration each episode per sample batch Sample a random leader policy $\pi_L^r$ each $o_L \in O_L$ Query $\pi_L^r$ for $\pi_L^r(o_L)$ Set context $\omega = \pi_L^r(o_L), o_L \in O_L$ each episode step Return $o_{L,t}$ to leader, $(\omega, o_{F,t})$ to follower Step environment using $a_{L,t} = \pi_L^r(o_L), a_{F,t} = \pi_F(\omega, o_{F,t})$ Update follower policy $\pi_F$ using collected sample batch using PG/PPO/DQN **Main Training Loop** Initialize leader policy $\pi_L$ each training iteration each episode per sample batch each $o_L \in O_L$ Query $\pi_L$ for $\pi_L(o_L)$ Set context $\omega = \pi_L(o_L), o_L \in O_L$ each episode step Return $o_{L,t}$ to leader, $(\omega, o_{F,t})$ to follower Step environment using $a_{L,t} = \pi_L^r(o_L), a_{F,t} = \pi_F(\omega, o_{F,t})$ Update leader policy $\pi_L$ using collected sample batch using PG/PPO/DQN

Table 3: Payoff Matrices used in the matrix-game experiments

| **Iterated Matrix Games (Figure 6 etc.)** | | |
|---|---|---|
| **Name** | **Leader Payoff** | **Follower Payoff** |
| prisoners dilemma | $\begin{pmatrix} -1 & -3 \\ 0 & -2 \end{pmatrix}$ | $\begin{pmatrix} -1 & 0 \\ -3 & -2 \end{pmatrix}$ |
| stag hunt | $\begin{pmatrix} 0 & -3 \\ -1 & -2 \end{pmatrix}$ | $\begin{pmatrix} 0 & -1 \\ -3 & -2 \end{pmatrix}$ |
| assurance | $\begin{pmatrix} 0 & -3 \\ -2 & -1 \end{pmatrix}$ | $\begin{pmatrix} 0 & -2 \\ -3 & -1 \end{pmatrix}$ |
| coordination | $\begin{pmatrix} 0 & -2 \\ -3 & -1 \end{pmatrix}$ | $\begin{pmatrix} 0 & -3 \\ -2 & -1 \end{pmatrix}$ |
| mixedharmony | $\begin{pmatrix} 0 & -1 \\ -3 & -2 \end{pmatrix}$ | $\begin{pmatrix} 0 & -3 \\ -1 & -2 \end{pmatrix}$ |
| harmony | $\begin{pmatrix} 0 & -1 \\ -2 & -3 \end{pmatrix}$ | $\begin{pmatrix} 0 & -2 \\ -1 & -3 \end{pmatrix}$ |
| noconflict | $\begin{pmatrix} 0 & -2 \\ -1 & -3 \end{pmatrix}$ | $\begin{pmatrix} 0 & -1 \\ -2 & -3 \end{pmatrix}$ |
| deadlock | $\begin{pmatrix} -2 & -3 \\ 0 & -1 \end{pmatrix}$ | $\begin{pmatrix} -2 & 0 \\ -3 & -1 \end{pmatrix}$ |
| prisoners delight | $\begin{pmatrix} -3 & -2 \\ 0 & -1 \end{pmatrix}$ | $\begin{pmatrix} -3 & 0 \\ -2 & -1 \end{pmatrix}$ |
| hero | $\begin{pmatrix} -3 & -1 \\ 0 & -2 \end{pmatrix}$ | $\begin{pmatrix} -3 & 0 \\ -1 & -2 \end{pmatrix}$ |
| battle | $\begin{pmatrix} -2 & -1 \\ 0 & -3 \end{pmatrix}$ | $\begin{pmatrix} -2 & 0 \\ -1 & -3 \end{pmatrix}$ |
| chicken | $\begin{pmatrix} -1 & -2 \\ 0 & -3 \end{pmatrix}$ | $\begin{pmatrix} -1 & 0 \\ -2 & -3 \end{pmatrix}$ |
| **Single-Shot Matrix Game (Appendix D)** | | |
| battle of the sexes | $\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ |
| **Modified Prisoner's Dilemma (Theorem 2)** | | |
| prisoners dilemma modified | $\begin{pmatrix} 0 & -2 \\ -1 & -3 \end{pmatrix}$ | $\begin{pmatrix} -1 & 0 \\ -3 & -2 \end{pmatrix}$ |

Table 4: 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 |

**Atari PPO**

| **Hyper-Parameter** | **Value** | **Hyper-Parameter** | **Value** |
|---|---|---|---|
| train_batch_size | 5000 | rollout_fragment_length | 100 |
| sgd_minibatch_size | 100 | num_sgd_iter | 10 |
| lambda | 0.95 | kl_coeff | 0.5 |
| clip_param | 0.1 | vf_clip_param | 10.0 |
| entropy_coeff | 0.01 | lr | 0.001 |
| num_rollout_workers | 10 | num_envs_per_worker | 5 |