Function Approximation for Solving Stackelberg Equilibrium in Large Perfect Information Games

Chun Kai Ling, J. Zico Kolter, Fei Fang

School of Computer Science, Carnegie Mellon University
chunkail@cs.cmu.edu, zkolter@cs.cmu.edu, feif@cs.cmu.edu

Abstract

Function approximation (FA) has been a critical component in solving large zero-sum games. Yet, little attention has been given towards FA in solving *general-sum* extensive-form games, despite them being widely regarded as being computationally more challenging than their fully competitive or cooperative counterparts. A key challenge is that for many equilibria in general-sum games, no simple analogue to the state value function used in Markov Decision Processes and zero-sum games exists. In this paper, we propose learning the *Enforceable Payoff Frontier* (EPF)—a generalization of the state value function for general-sum games. We approximate the optimal *Stackelberg extensive-form correlated equilibrium* by representing EPFs with neural networks and training them by using appropriate backup operations and loss functions. This is the first method that applies FA to the Stackelberg setting, allowing us to scale to much larger games while still enjoying performance guarantees based on FA error. Additionally, our proposed method guarantees incentive compatibility and is easy to evaluate without having to depend on self-play or approximate best-response oracles.

1 Introduction

A central challenge in modern game solving is to handle large game trees, particularly those too large to traverse or even specify. These include board games like Chess, Poker (Silver et al. 2018, 2016; Brown and Sandholm 2017, 2019; Moravčík et al. 2017; Bakhtin et al. 2021; Gray et al. 2021) and modern video games with large state and action spaces (Vinyals et al. 2019). Today, scalable game solving is frequently achieved via function approximation (FA), typically by using neural networks to model state values and harnessing the network's ability to generalize its evaluation to states never encountered before (Silver et al. 2016, 2018; Moravčík et al. 2017; Schmid et al. 2021). Methods employing FA have achieved not only state-of-the-art performance, but also exhibit more human-like behavior (Kasparov 2018).

Surprisingly, FA is rarely applied to solution concepts used in general-sum games such as Stackelberg equilibrium, which are generally regarded as being more difficult to solve than the perfectly cooperative/competitive Nash equilibrium. Indeed, the bulk of existing literature centers around on methods such as exact backward induction (Bosanský et al. 2015; Bošanskỳ et al. 2017), incremental strategy generation (Černỳ, Bošanskỳ, and Kiekintveld 2018; Cermak et al. 2016; Karwowski and Mańdziuk 2020), and mathematical programming (Bosansky and Cermak 2015).[1] While exact, these methods rarely scale to large game trees, especially those too large to traverse, severely limiting our ability to tackle general-sum games that are of practical interest, such as those in security domains like wildlife poaching prevention (Fang et al. 2017) and airport patrols (Pita et al. 2008). For non-Nash equilibrium in general-sum games, the value of a state often *cannot* be summarized as a scalar (or fixed sized vector), rendering the direct application of FA-based zero-sum solvers like (Silver et al. 2018) infeasible.

In this paper, we propose applying FA to model the *Enforceable Payoff Frontier* (EPF) for each state and using it to solve for the *Stackelberg extensive-form correlated equilibrium* (SEFCE) in two-player games of perfect information. Introduced in (Bošanskỳ et al. 2017; Bosanský et al. 2015; Letchford and Conitzer 2010), EPFs capture the tradeoff between player payoffs and is analogous to the state value in zero-sum games.[2] Specifically, we (i) study the pitfalls that can occur with using FA in general-sum games, (ii) propose a method for solving SEFCEs by modeling EPFs using neural networks and minimizing an appropriately designed Bellman-like loss, and (iii) provide guarantees on incentive compatibility and performance of our method. Our approach is the first application of FA in Stackelberg settings without relying on best-response oracles for performance guarantees. Experimental results show that our method can (a) approximate solutions in games too large to explicitly traverse, and (b) generalize learned EPFs over states in a repeated setting where game payoffs vary based on features.

2 Preliminaries and Notation

A 2-player *perfect information* game G is represented by a finite game tree with game states $s \in \mathcal{S}$ given by vertices and action space $\mathcal{A}(s)$ given by directed edges starting from s.

[1] Meta-game solving (Lanctot et al. 2017; Wang et al. 2019) is used in zero-sum games, but not general-sum Stackelberg games.

[2] The idea of an EPF was initially used by (Letchford and Conitzer 2010) to give a polynomial time solution for SSEs. However, they (as well as other work) do not propose any naming.

Each state belongs to either player P_1 or P_2; we denote these disjoint sets by $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively. Every leaf (terminal state) $\ell \in \mathcal{L} \subseteq \mathcal{S}$ of G is associated with payoffs, given by $r_i(\ell)$ for each player i. Taking action $a \in \mathcal{A}(s)$ at state $s \notin \mathcal{L}$ leads to $s' = T(a; s)$, where $s' \in \mathcal{S}$ is the next state and T is the deterministic transition function. Let $\mathcal{C}(s) = \{s' \mid T(a; s) = s', a \in \mathcal{A}(s)\}$ denote the immediate children of s. We say that state s precedes ($\sqsubset$) state s' if $s \neq s'$ and s is an ancestor of s' in G, and write $\sqsubseteq$ if allowing $s = s'$. An action $a \in \mathcal{A}(s)$ *leads to* s' if $s \sqsubset s'$ and $T(a; s) \sqsubseteq s'$. With a slight abuse of notation, we denote $T(a; s) \sqsubseteq s'$ by $a \sqsubset s'$ or $(s, a) \sqsubset s'$. Since G is a tree, for states s, s' where $s \sqsubset s'$, exactly one $a \in \mathcal{A}(s)$ such that $(s, a) \sqsubseteq s'$. We use the notation $\sqsupseteq$ and $\sqsupset$ when the relationships are reversed.

A strategy π_i, where $i \in \{\mathsf{P}_1, \mathsf{P}_2\}$, is a mapping from state $s \in \mathcal{S}_i$ to a distribution over actions $\mathcal{A}(s)$, i.e., $\sum_{a \in \mathcal{A}(s)} \pi_i(a; s) = 1$. Given strategies π_1 and π_2, the probability of reaching $\ell \in \mathcal{L}$ starting from s is given by $p(\ell | s; \pi_1, \pi_2) = \prod_{i \in \{\mathsf{P}_1, \mathsf{P}_2\}} \prod_{(s', a); s \sqsubseteq s', (s', a) \sqsubset \ell, s' \in \mathcal{S}_i} \pi_i(a; s')$, and player i's expected payoff starting from s is $R_i(s; \pi_1, \pi_2) = \sum_{\ell \in \mathcal{L}} p(\ell | s; \pi_1, \pi_2) r_i(\ell)$. We use as shorthand $p(\ell; \pi_1, \pi_2)$ and $R_i(\pi_1, \pi_2)$ if s is the root. A strategy π_2 is a *best response* to a strategy π_1 if $R_2(\pi_1, \pi_2) \geq R_2(\pi_1, \pi_2')$ for all strategies π_2'. The set of best responses to π_1 is written as $BRS_2(\pi_1)$.

The *grim strategy* $\arg\min_{\pi_1} \max_{\pi_2} R_2(\pi_1, \pi_2)$ of P_1 towards P_2 is one which guarantees the lowest payoff for P_2. Conversely, the *joint altruistic strategy* $\arg\max_{\pi_1, \pi_2} R_2(\pi_1, \pi_2)$ is one which maximizes P_2's payoff. We restrict grim and altruistic strategies to those which are subgame-perfect, i.e., they remain the optimal if the game was rooted at some other state.[3] Grim and altruistic strategies ignore P_1's own payoffs and can be computed by backward induction. For each state, we denote by $\underline{V}(s)$ and $\overline{V}(s)$ the internal values of P_2 for grim and altruistic strategies obtained via backward induction.

2.1 Stackelberg Equilibrium in Perfect Information Games

In a Strong Stackelberg equilibrium (SSE), there is a distinguished *leader* and *follower*, which we assume are P_1 and P_2 respectively. The leader *commits* to any strategy π_1 and the follower best responds to the leader, breaking ties by selecting $\pi_2 \in BRS_2(\pi_1)$ such as to benefit the leader. [4] Solving for the SSE entails finding the optimal commitment for the *leader*, i.e., a pair $\pi = (\pi_1, \pi_2)$ such that $\pi_2 \in BRS_2(\pi_1)$ and $R_1(\pi_1, \pi_2)$ is to be maximized.

It is well-known that the optimal SSE will perform no worse (for the leader) than Nash equilibrium, and often much better. Consider the game in Figure 1a with $k_1 = k_2 = 0$. If the expected follower payoff from staying is less than 0, then it would exit immediately. Hence, solutions such as the subgame perfect Nash gives a leader payoff of 0. The optimal Stackelberg solution is for the leader to commit to a uniform strategy—this ensures that staying yields the follower a payoff of 0, which under the tie-breaking assumptions of SSE nets the leader a payoff of 4.5.

Stackelberg Extensive-Form Correlated Equilibrium For this paper, we will focus on a relaxation of the SSE known as the Stackelberg extensive-form correlated equilibirum (SEFCE), which allows the leader to explicitly recommend actions to the follower *at the time of decision making*. If the follower deviates from the recommendation, the leader is free to retaliate—typically with the grim strategy. In a SEFCE, P_1 takes and recommends actions to maximize its reward, subject to the constraints that the recommendations are sufficiently appealing to P_2 relative to threat of P_2 facing the grim strategy after any potential deviation.

Definition 1 (Minimum required incentives)**.** *Given $s \in \mathcal{S}_2$, $s' \in \mathcal{C}(s)$, we define the minimum required incentive $\tau(s') = \max_{s^! \in \mathcal{C}(s); s^! \neq s'} \underline{V}(s^!)$, i.e., the minimum amount that P_1 needs to promise P_2 under s' for it to be reached.*

Definition 2 (SEFCE)**.** *A strategy pair $\pi = (\pi_1, \pi_2)$ is a SEFCE if it is incentive compatible, i.e., for all $s \in \mathcal{S}_2$, $a \in \mathcal{A}(s)$, $\pi_2(a; s) > 0 \implies R_2(T(a; s); \pi_1, \pi_2) \geq \tau(T(a; s))$. Additionally, π is optimal if $R_1(\pi_1, \pi_2)$ is maximized.*

In Section 3, we describe how optimal SEFCE can be computed in polynomial time for perfect information games.

2.2 Function Approximation of State Values

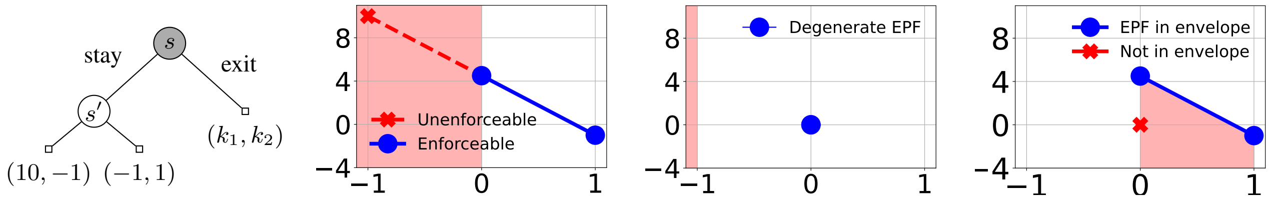
When finding Nash equilibrium in perfect information games, the *value* v_s of a state is a crucial quantity which summarizes the utility obtained from s onward, assuming optimal play from all players. It contains sufficient information for one to obtain an optimal solution after using them to 'replace' subtrees. Typically v_s should only rely on states $s' \sqsupseteq s$. In zero-sum games, $v_s = \underline{V}_s$ while in cooperative games, $v_s = \overline{V}_s$. Knowing the true value of each state immediately enables the optimal policy via one-step lookahead. While v_s can be computed over all states by backward induction, this is not feasible when G is large. A standard workaround is to replace v_s with an approximate $\tilde{v}_s$ which is then used in tandem with some search algorithm (depth-limited search, Monte-Carlo tree search, etc.) to obtain an approximate solution. Today, $\tilde{v}_s$ is often *learned*. By representing $\tilde{v}$ with a rich function class over state features (typically using a neural network), modern solvers are able to generalize $\tilde{v}$ across large state spaces without explicitly visiting every state, thus scaling to much larger games.

Fitted Value Iteration. A class of methods closely related to ours is Fitted Value Iteration (FVI) (Lagoudakis and Parr 2003; Dietterich and Wang 2001; Munos and Szepesvári 2008). The idea behind FVI is to optimize for parameters such as to minimize the *Bellman loss* over sampled states by treating it as a regular regression problem. [5] Here, the

[3] This is to avoid strategies which play arbitrarily at states which have 0 probability of being reached.

[4] Commitment rights are justified by repeated interactions. If the P_1 reneges on its commitment, P_2 plays another best response, which is detrimental to the leader. This setting is unlike (De Jonge and Zhang 2020) which uses binding agreements.

[5] We distinguish RL and FVI in that the transition function is known explicitly and made used of in FVI.



(a) Toy example. (b) EPF at vertex s'. (c) EPF at vertex after exiting. (d) EPF at root vertex s.

Figure 1: (a) Game tree to illustrate computation of SEFCE. Leader ○, follower ● and leaf □ states are vertices and edges are actions. (b-d) EPFs at s', after exiting and s. The x and y axes are follower (μ_2) and leader payoffs ($U_s(\mu)$). In (b) the pink regions give P_2 too little and are truncated. In (d), the pink regions are not part of the upper concave envelope and removed.

Bellman loss measures the distance between $\tilde{v}_s$ and the estimated value using one-step lookahead using $\tilde{v}$. If this distance is 0 for all s, then $\tilde{v}$ matches the optimal v. In practice, small errors in FA accumulate and cascade across states, lowering performance. Thus, it is important to bound performance as a function of the Bellman loss over all s.

2.3 Related Work

Some work has been done in generalizing state values in general-sum games, but few involve learning them. Related to ours is (Murray and Gordon 2007; MacDermed et al. 2011; Dermed and Charles 2013), which approximate the achievable set of payoffs for correlated equilibrium, and eventually SSE (Letchford et al. 2012) in stochastic games. These methods are analytical in nature and scale poorly. (Pérolat et al. 2017; Greenwald et al. 2003) propose a Q-learning-like algorithm over general-sum Markov games, but do not apply FA and only consider stationary strategies which preclude strategies involving long range threats like the SSE. (Zinkevich, Greenwald, and Littman 2005) show a class of general-sum Markov games where value-iteration like methods will necessarily fail. (Zhong et al. 2021) study reinforcement learning in the Stackelberg setting, but only consider followers with myopic best responses. (Castelletti, Pianosi, and Restelli 2011) apply FVI in a multiobjective setting, but do not consider the issue of incentive compatibility. Another approach is to apply reinforcement learning and self-play (Leibo et al. 2017). Recent methods account for the nonstationary environment each player faces during training (Foerster et al. 2017; Perolat et al. 2022); however they have little game theoretical guarantees in terms of incentive compatibility, particularly in non zero-sum games.

3 Review: Solving SEFCE via Enforceable Payoff Frontiers

In Section 2, we emphasized the importance of the value function v in solving zero-sum games. In this section, we review the analogue for SEFCE in the general-sum games, which we term as *Enforceable Payoff Frontiers* (EPF). Introduced in (Letchford and Conitzer 2010), the EPF at state s is a *function* $U_s : \mathbb{R} \mapsto \mathbb{R} \cup \{-\infty\}$, such that $U_s(\mu_2)$ gives the maximum leader payoff for a SEFCE for a game rooted at s, on condition that P_2 obtains a payoff of μ_2. All leaves $s \in \mathcal{L}$ have degenerate EPFs $U_s(r_2(s)) = r_1(s)$ and $-\infty$ everywhere else. EPFs capture the tradeoff in payoffs between P_1 and P_2, making them useful for solving SEFCEs. We now review the two-phase algorithm of (Bošanskỳ et al. 2017) using the example game in Figure 1a with $k_1 = k_2 = 0$. This approach forms the basis for our proposed FA method.

Phase 1: Computing EPF by Backward Induction. The EPF at s' is given by the line segment connecting payoffs of its children EPF and $-\infty$ everywhere else. This is because the leader is able to freely mix over actions. To compute U_s, we consider in turn the EPFs after staying or exiting. Case 1: P_1 is recommending P_2 to stay. For incentive compatibility, it needs to *promise* P_2 a payoff of at least 0 under $T(\text{stay}; s) = s'$. Thus, we **left-truncate** the regions of the EPF at s' which violate this promise, leaving behind the blue segment (Figure 1b), which represents the payoffs at s' that are *enforceable* by P_1. Case 2: P_1 is recommending P_2 to exit. To discourage P_2 from staying, it commits to the grim strategy at s' if P_2 chooses to stay instead, yielding P_2 a payoff of $-1 \leq k_2 = 0$. Hence, no truncation is needed and the set of enforceable payoffs is the (degenerate) blue line segment (Figure 1c). Finally, to recover U_s, observe that we can achieve payoffs on any line segment connecting point across the EPFs of s's children. This union of points on such lines (ignoring those leader-dominated) is given by the **upper concave envelope** of the blue segments in Figure 1b and 1c; this removes $\{(0, 0)\}$, giving the EPF in Figure 1d.

More generally, let g_1 and g_2 be functions such that $g_j : \mathbb{R} \mapsto \mathbb{R} \cup \{-\infty\}$. We denote by $g_1 \bigwedge g_2$ their upper concave envelope, i.e., $\inf\{h(\mu) \mid h \text{ is concave and } h \geq \max\{g_1, g_2\} \text{ over } \mathbb{R}\}$. Since $\bigwedge$ is associative and commutative, we use as shorthand $\bigwedge_{\{\cdot\}}$ when applying $\bigwedge$ repeatedly over a finite set of functions. In addition, we denote $g \triangleright t$ as the left-truncation of the g with threshold $t \in \mathbb{R}$, i.e., $[g \triangleright t](\mu) = g(\mu)$ if $\mu \geq t$ and $-\infty$ otherwise. Note that both $\bigwedge$ and $\triangleright$ are closed over concave functions. For any $s \in \mathcal{S}$, its EPF U_s can be concisely written in terms of its children EPF $U_{s'}$ (where $s' \in \mathcal{C}(s)$) using $\bigwedge$, $\triangleright$ and $\tau(s')$.

$$U_s(\mu) = \begin{cases} \left[\bigwedge_{s' \in \mathcal{C}(s)} U_{s'}\right](\mu) & \text{if } s \in \mathcal{S}_1 \\ \left[\bigwedge_{s' \in \mathcal{C}(s)} U_{s'} \triangleright \tau(s')\right](\mu) & \text{if } s \in \mathcal{S}_2 \end{cases}, \quad (1)$$

which we apply in a bottom-up fashion to complete Phase 1.

Phase 2: Extracting Strategies from EPF. Once U_s has been computed for all $s \in \mathcal{S}$, we can recover the optimal strategy π_1 by applying one-step lookahead starting from the root. First, we extract $(\mathsf{OPT}_2, \mathsf{OPT}_1)$, the coordinates of the maximum point in U_{root}, which contain payoffs under the optimal π. Here, this is $(0, 4.5)$. We initialize $\mu_2 = \mathsf{OPT}_2$, which represents P_1's promised payoff to P_2 at the current state s. Next, we traverse G depth-first. By construction, $U_s(\mu_2) > -\infty$ and the point $(\mu_2, U_s(\mu_2))$ is the convex combination of either 1 or 2 points belonging to its children EPFs. The mixing factors correspond to the optimal strategy $\pi(a; s)$. If there are 2 distinct children s', s'' with mixing factor α', α'', we repeat this process for s', s'' with $\mu_2' = \mu_2/\alpha', \mu_2'' = \mu_2/\alpha''$, otherwise we repeat the process for s' and $\mu_2' = \mu_2$. For our example, we start at s, $\mu_2 = 0$, which was obtained by P_2 playing 'stay' exclusively, so we keep μ_2 and move to s'. At s', $\mu = 0$ by mixing uniformly, which gives us the result in Section 2.

Theorem 1 ((Bošanskỳ et al. 2017; Bosanský et al. 2015))**.** *(i) U_s is piecewise linear concave with number of knots*[6] *no greater than the number of leaves beneath s. (ii) Using backward induction, SEFCEs can be computed in polynomial time (in $|\mathcal{S}|$) even in games with chance. EPFs continue to be piecewise linear concave.*

Markovian Property. Just like state values v_s in zero-sum games, we can replace any internal vertex s in G with its EPF while not affecting the optimal strategy in all other branches of the game. This can done by adding a single leader vertex with actions leading to terminal states with payoffs corresponding to the knots of U_s. Since U_s is obtained via backward induction, it only depends on states beneath s. In fact, if two games G and G' (which could be equal to G) shared a common subgame rooted in s and s' respectively, we could reuse the U_s found in G for $U_{s'}$ in G'. This observation underpins the inspiration for our work—if s and s' are similar in some features, then U_s and $U_{s'}$ are likely similar and it should be possible to *learn* and *generalize* EPFs over states.

4 Challenges in Applying FA to EPF

We now return to our original problem of applying FA to find SEFCE. Our idea, outlined in Algorithm 1 and 2 is a straightforward extension of FVI. Suppose each state has features $f(s)$—in the simplest case this could be a state's history. We design a neural network $E_\phi(f)$ parameterized by ϕ. This network maps state features $f(s)$ to some representation of $\tilde{U}_s$, the approximated EPFs. To achieve a good approximation, we optimize ϕ by minimizing an appropriate Bellman-like loss (over EPFs) based on Equation (1) while using our approximation $\tilde{U}_s$ in lieu of U_s. Despite its simplicity, there remain several design considerations.

Algorithm 1: Training Pipeline

```
Sample trajectory $s^{(1)}_{\text{new}}, \ldots, s^{(t)}_{\text{new}}$
Update replay buffer $\mathcal{B}$ with $s^{(1)}, \ldots, s^{(t)}$
for $i \in \{1, \ldots, t\}$ do
    Sample batch $S = \{s^{(1)}, \ldots s^{(n)}\} \subseteq \mathcal{B}$
    $\ell \leftarrow$ ComputeLoss($S; E_\phi$)
    Update $\phi$ using $\partial\ell/\partial\phi$
```

Algorithm 2: ComputeLoss($S; E_\phi$)

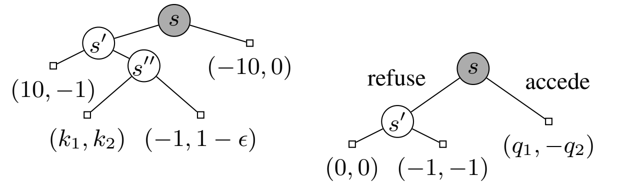
```
for $i \in \{1 \ldots n\}$ do
    $\tilde{U}_{s^{(i)}} \leftarrow E_\phi(f(s^{(i)}))$
    $\tilde{U}_{s^{(j)}_{\text{next}}} \leftarrow E_\phi(f(s^{(j)}_{\text{next}}))$    for all $s^{(j)}_{\text{next}} \in \mathcal{C}(s^{(i)})$
    Compute $\tilde{U}^{\text{target}}_{s^{(i)}}$ using Equation (1) and $\{\tilde{U}_{s^{(j)}_{\text{next}}}\}$
return $\sum_i L(\tilde{U}_{s^{(i)}}, \tilde{U}^{\text{target}}_{s^{(i)}})$
```

EPFs are the 'right' object to learn. Unlike state values, representing an exact EPF at a state s could require more than constant memory since the number of knots could be linear in the number of leaves underneath it (Theorem 1). Can we get away with summarizing a state with a scalar or a small vector? Unfortunately, any 'lossless summary' which enjoys the Markovian property necessarily encapsulates the EPF. To see why, consider the class of games G_k in Figure 1a with $k_1 = -2$ and $k = k_2 \in [-1, 1]$. The optimal leader payoff for any G_k is $\frac{9-11k}{2}$, which is precisely $U_{s'}(k)$ (Figure 1b). Now consider any lossless summary for s' and use it to solve *every* G_k. The resultant optimal leader payoffs can recover $U_{s'}(\mu_2)$ between $\mu_2 \in [-1, 1]$. This implies that no lossless summary more compact than the EPF exists.

Unfulfillable Promises Arising from FA Error. Consider the game in Figure 2a with $k_1 = -10$, $k_2 = -1$. The exact $U_{s'}$ is the line segment combining the points $(-1, 10)$ and $(1-\epsilon, -1)$, shown in green in Figure 3a. However, let us suppose that due to function approximation we instead learned the blue line segment containing $(-1, 10)$ and $(1, -1)$. Performing Phase 2 using $\tilde{U}$, the policy extracted at s' is once again the uniform policy and requires us to promise the follower a utility of 1 in s''. However, achieving a payoff of 1 is *impossible* regardless of how much the leader is willing to sacrifice, since the maximum outcome under s'' is $1 - \epsilon$. Since this is an *unfulfillable promise*, the follower's best responds by exiting in s, which gives the leader a payoff of -10. In general, unfulfillable promises due to small FA error can lead to arbitrarily low payoffs. In fact, one could argue that $\tilde{U}$ does not even *define* a valid policy.

Costly Promises. Consider the case where $k_1 = -30, k_2 = 1$ while keeping $\tilde{U}_{s'}$ the same. Here, the promise of 1 at s'' *is* fulfillable, but involves incurring a cost of -30, which is even lower than having follower staying (Figure 3b). In general, this problem of *costly promises* stems from the EPF being wrongly estimated, even for a small range of μ_2. We can see how costly promises arise even from small ϵ is. The underlying issue is that in general, U_s can have large Lipschitz constants (e.g., proportionate to $(\max_s r_1(s) - \min_s r_1(s)) / (\min |r_2(s) - r_2(s)|)$). The existence of costly payoffs rules out EPF representations based

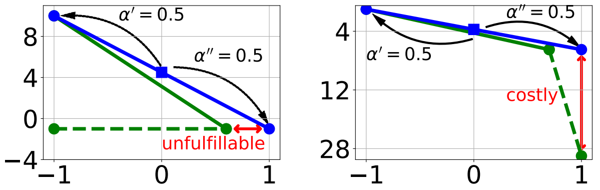
[6] Knots are where the slope of the EPF changes.



(a) Game tree to illustrate unfulfillable and costly promises.

(b) Stage game used in the TANTRUM game.

Figure 2: Games used in Sections 4 and 6. Leader ○, follower ● and leaf □ states are vertices and edges are actions.



(a) Unfulfillable promise.

(b) Costly promise.

Figure 3: EPFs for (a) unfulfillable promises and (b) costly promises. Blue lines are estimated EPFs $\tilde{U}_{s'}$, solid and dotted green lines are true EPFs $U_{s'}$, $U_{s''}$. In both cases, FA error leads us to believe that the payoff given by the blue square at $(0, 4.5)$ can be achieved by mixing the endpoints of $\tilde{U}_{s'}$ with probability $\alpha' = \alpha'' = 0.5$ (black curves).

on discretizing the space of μ_2, since small errors incurred by discretization could lead to huge drops in performance.

5 FA of EPF with Performance Guarantees

We now design our method using the insights from Section 4. We learn EPFs without relying on discretization over P_2 payoffs μ_2. Unfulfillable promises are avoided entirely by ensuring that the set of μ_2 where $\tilde{U}_s(\mu_2) > -\infty$ lies within some known set of achievable P_2 payoffs, while costly promises are mitigated by suitable loss functions.

Representing EPFs using Neural Networks. Our proposed network architecture represents EPFs by a small set of $m \geq 2$ points $P_\phi(s) = \{(x_j, y_j)\}$, for $j \in [m]$. Here, m is a hyperparameter trading off complexity of the neural network E_ϕ with its representation power. The approximated EPF $\tilde{U}_s$ is the linear interpolation of these m points; and $\tilde{U}_s = -\infty$ if $\mu_2 > \max_j x_j$ or $\mu_2 < \min_j x_j$. For now, we make the assumption that follower payoffs under the altruistic and grim strategy ($\overline{V}(s)$ and $\underline{V}(s)$) are known *exactly* for all states. Through the architecture of E_ϕ that for all $j \in [m]$, we have $\underline{V}(s) \leq x_j \leq \overline{V}(s)$. As we will see, this helps avoid unfulfillable promises and allows for convenient loss functions.

Concretely, our network $E_\phi(f(s); \underline{V}(s), \overline{V}(s))$ takes in as inputs state features $f(s)$, lower and upper bounds $\underline{V}(s) \leq \overline{V}(s)$ and outputs a matrix in $\mathbb{R}^{m \times 2}$ representing $\{(x_j, y_j)\}$ where $x_1 = \underline{V}(s)$ and $x_m = \overline{V}(s)$. For simplicity, we use a multilayer feedforward network with depth d, width w and ReLU activations for each layer. Serious applications should utilize domain specific architectures. Denoting the output of the last fully connected layer by $h^{(d)}(f(s)) \in \mathbb{R}^w$, for $j \in \{2 \ldots m-1\}$ and $k \in [m]$ we set

$$x_j = \sigma\left(z_{x,j}^T h^{(d)}(f(s)) + b_{x,j}\right) \cdot \left(\overline{V}(s) - \underline{V}(s)\right) + \underline{V}(s),$$
$$y_k = z_{y,k}^T h^{(d)}(f(s)) + b_{y,k},$$

and $x_1 = \underline{V}(s)$ and $x_m = \overline{V}(s)$, where $\sigma(x) = 1/(1 + \exp(-x))$. Here, $z_{x,j}, z_{y,k} \in \mathbb{R}^w$ and $b_{x,j}, b_{y,k} \in \mathbb{R}$ are weights and biases, which alongside the parameters from feedforward network form the network parameters ϕ to be optimized. Since $\tilde{U}_s$ is represented by its knots (given by $P_\phi(s)$), $\bigwedge$ and consequently, (1) may be performed *explicitly* and *efficiently*, returning an entire EPF represented by its knots (as opposed to the EPF evaluated at a single point). This is crucial, since the computation is performed every state every iteration (Line 4 of Algorithm 2).

Loss Functions for Learning EPFs. Given 2 EPFs $\tilde{U}_s, \tilde{U}'_s$ we minimize the following loss to mitigate costly promises,

$$L_\infty(\tilde{U}_s, \tilde{U}'_s) = \max_{\mu_2} |\tilde{U}_s(\mu_2) - \tilde{U}'_s(\mu_2)|.$$

L_∞ was chosen specifically to incur a large loss if the approximation is wildly inaccurate in a small range of μ_2 (e.g., Figure 3b). Achieving a small loss requires that $\tilde{U}_s(\mu_2)$ approximates $\tilde{U}'_s(\mu_2))$ well for all μ_2. This design decision is particularly important. For example, contrast L_∞ with another intuitive loss $L_2(\tilde{U}_s, \tilde{U}'_s) = \int_{\mu_2} (\tilde{U}_s(\mu_2) - \tilde{U}'_s(\mu_2))^2 d\mu_2$. Observe that L_2 is exceedingly small in the example of Figure 3b — in fact, when ϵ is small enough leads to almost no loss, even though the policy as discussed in Section 4 is highly suboptimal. This phenomena leads to costly promises, which was indeed observed in our tests.

Our Guarantees. Any learned $\tilde{U}$ implicitly defines a policy $\tilde{\pi}$ by one-step lookahead using Equation (1) and the method described in Phase 2 (Section 3). Extracting $\tilde{\pi}$ need not be done offline for all $s \in \mathcal{S}$; in fact, when G is too large it is necessary that we only extract $\tilde{\pi}(\cdot; s)$ on-demand. Nonetheless, $\tilde{\pi}$ enjoys some important properties.

Theorem 2 (Incentive Compatibility)**.** *For any policy $\tilde{\pi}$ obtained using our method, any $s \in \mathcal{S}_2$ and $a \in \mathcal{A}(s)$, we have $\tilde{\pi}_s(a; s) > 0 \implies R_2(T(a; s); \tilde{\pi}) \geq \tau(T(a; s))$.*

Theorem 3 (FA Error)**.** *If $L_\infty(\tilde{U}_s, \tilde{U}_s^{\text{target}}) \leq \epsilon$ for all $s \in \mathcal{S}$, then $|R_1(\tilde{\pi}) - R_1(\pi^*)| = \mathcal{O}(D\epsilon)$ where D is the depth of G and π^* is the optimal strategy.*

Here, $T(a; s)$ is transition function (Section 2). Recall from Section 2 that for π to be an optimal SEFCE, we require (i) incentive compatibility and (ii) $R_1(\pi)$ to be maximized. Theorems 2 and 3 illustrate how our approach disentangles these criteria. Theorem 2 guarantees that P_2 will always be incentivized to follow P_1's recommendations, i.e., there will be no unexpected outcomes arising from unfulfillable promises. Crucially, this is a hard constraint which is satisfied solely due to our choice of network architecture, which ensures that $\tilde{U}_s(\mu_2) = -\infty$ when $\mu_2 > \overline{V}_s$ for *any* $\tilde{\pi}$ obtained from $\tilde{U}$. Conversely, Theorem 3 shows that the goal of maximizing R_1 *subject to incentive compatibility* is achieved by attaining a small FA error across all states. This distinction is important. Most notably, incentive compatibility is no longer dependent on convergence during training. This *explicit* guarantee stands in contrast with methods employing self-play reinforcement learning agents; there, incentive compatibility follows *implicitly* from the apparent convergence of a player's strategy. This guarantee has practical implications, for example, evaluating the quality of $\tilde{\pi}$ can be done by estimating $R_1(\tilde{\pi})$ based on sampled trajectories, while implicit guarantees requires incentive compatibility to be demonstrated using some approximate best-response oracle and usually involves expensive training of a RL agent.

The primary limitation of our method is when $\overline{V}$ and $\underline{V}$ (and hence τ) are not known exactly. As it turns out, we can instead use upper and lower approximations while still retaining incentive compatibility. Let $\tilde{\pi}_1^{\text{grim}}$ be an *approximate grim strategy*. Define $\underset{\sim}{V}(s)$ to be the expected follower payoffs at s when faced best-responding to $\tilde{\pi}_1^{\text{grim}}$, i.e., $R_2(s; \tilde{\pi}_1^{\text{grim}}, \pi_2)$, where $\pi_2 \in BRS_2(\tilde{\pi}_1^{\text{grim}})$. Following Definition 1, the *approximate minimum required incentive* is $\tilde{\tau}(s') = \max_{s^! \in \mathcal{C}(s); s^! \neq s'} \underset{\sim}{V}(s^!)$ for all $s \in \mathcal{S}_2$, $s' \in \mathcal{C}(s)$. Similarly, let $\tilde{\pi}^{\text{alt}}$ be an *approximate joint altruistic strategy* and its resultant internal payoffs in each state be $\tilde{V}(s)$.

Under the mild assumption that $\tilde{\pi}^{\text{alt}}$ always benefits P_2 more than the $\tilde{\pi}_1^{\text{grim}}$, i.e., $\tilde{V}(s) \geq \underset{\sim}{V}(s)$ for all s, we can replace the $\tau, \underline{V}$ and $\overline{V}$ with $\tilde{\tau}, \underset{\sim}{V}$ and $\tilde{V}$ and maintain incentive compatibility (Theorem 2). The intuition is straightforward: if P_2's threats are 'good enough', parts of the EPF will still be enforceable. Furthermore, promises will always be fulfillable since EPFs domains are now limited to be no greater than $\tilde{V}(s)$, which we know can be achieved by definition. Unfortunately, Theorem 3 no longer holds, not even in terms of $\max_s |\tilde{\tau}(s) - \tau(s)|$. This is again due to the large Lipschitz constants of U_s. However, we have the weaker guarantee (whose proof follows that of Theorem 3) that performance is close to that predicted at the root.

Theorem 4 (FA Error with Weaker Bounds)**.** *If* $L_\infty(\tilde{U}_s, \tilde{U}_s^{\textit{target}}) \leq \epsilon$ *for all* $s \in \mathcal{S}$, *then* $|R_1(\tilde{\pi}) - \widetilde{\mathsf{OPT}}_2| = \mathcal{O}(D\epsilon)$ *where* D *is the depth of* G *and* $\widetilde{\mathsf{OPT}}_2 = \max_{\mu_2} \tilde{U}_{\textit{root}}(\mu_2)$.

Remark. The key technical difficulty here is finding $\tilde{V}$. In our experiments, $\tilde{\pi}_1^{\text{grim}}$ can be found analytically. In general large games, we can approximate $\tilde{\pi}_1^{\text{grim}}, \tilde{V}$ by searching over $\mathcal{S}_2$, but use heuristics when expanding nodes in $\mathcal{S}_1$.

Implementation Details. (i) We use several techniques typically used to stabilize training such as target networks (Arulkumaran et al. 2017; Mnih et al. 2015) and prioritized experience replay (Schaul et al. 2015). (ii) In practice, instead of L_∞, we found it easier to train a loss based on the sum of the squared distances at the x-coordinate of the knots in $\tilde{U}_s$ and $\tilde{U}'_s$, i.e., $L = \sum_{\mu_2 \in \{\text{knots}\}} [\tilde{U}_s(\mu_2) - \tilde{U}'_s(\mu_2)]^2$. Since L upper bounds L_∞^2, using it also avoids costly promises and allows us to enjoys a similar FA guarantee. (iii) If G has a branching factor of β, then (1) in Algorithm 2 can be executed in $\mathcal{O}(\beta m)$ time. In practice, we use a brute force method better suited for batch GPU operations which runs in $\mathcal{O}((\beta m)^3)$. (iv) We train using only the decreasing portions of $\tilde{U}_s$. This does not lead to any loss in performance since payoffs in the increasing portion of an $\tilde{U}_s$ are Pareto dominated. We do not want to 'waste' knots on learning the meaningless increasing portion. (v) Training trajectories were obtained by taking actions uniformly at random. Specifics for all implementation details are in the Appendix.

6 Experiments

We focused on the following two synthetic games. Game details and experiment environments are in the Appendix. Code is at https://github.com/lingchunkai/learn-epf-sefce.

Tantrum. TANTRUM is the game in Figure 2b repeated n times, with $q_1 > 0, q_2 \geq 1$, and rewards accumulated over stages. The only way P_1 can get positive payoffs is by threatening to throw a trantrum with the mutually destructive $(-1, -1)$ outcome. Since $q_2 > 1$, P_2 has to use threats spanning over stages to sufficiently entice P_2 to accede. Even though TRANTRUM has $\mathcal{O}(3^n)$ leaves, it is clear that the grim (resp. altruistic) strategy is to throw (resp. not throw) a tantrum at every step. Hence $\overline{V}$ and $\underline{V}$ are known even when n is large, making TANTRUM a good testbed. The raw features $f(s)$ is a 5-dimensional vector, the first 3 are the occurrences count of outcomes for previous stages, and the last 2 being a one-hot vector indicating the current state.

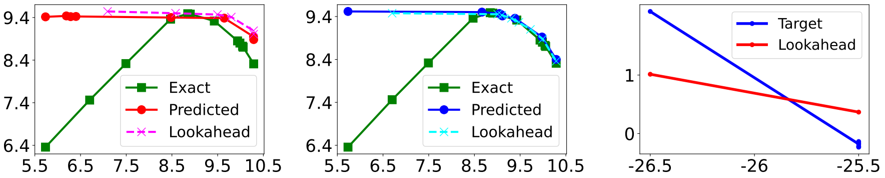
Resource Collection. RC is played on a $J \times J$ grid with a time horizon n. Each cell contains varying quantities of 2 different resources $\mathsf{r}_1(x, y), \mathsf{r}_2(x, y) \geq 0$, both of which are collected (at most once) by either players entering. Players begin in the center and alternately choose to either move to an adjacent cell or stay put. Each P_i is only interested in resource i, and players agree to pool together resources when the game ends. RC gives P_1 the opportunity to threaten P_2 with going 'on strike' if P_2 does not move to the cells that P_1 recommends. RC has approximately $\mathcal{O}(25^n)$ leaves. The grim strategy is for P_1 to stay put. However, unlike TANTRUM, computing $\underline{V}$ and $\overline{V}$ still requires search (at least for P_2) at each state, which is still computationally expensive. We use as features (a) one-hot vector showing past visited locations, (b) the current coordinates of each player and whose turn it is (c) the amount of each resource collected, and (d) the number of rounds remaining.

6.1 Experimental Setup

Games with Fixed Parameters. We run 3 sub-experiments. [**RC**] We experimented with RC with $J = 7, n = 4$ over 10 different games. Rewards r_i were generated using a log-Gaussian process over (x, y) to simulate spatial correlations (details in Appendix). We also report the payoffs from a 'non-strategic' P_1 which optimizes only for resources it collects, while letting P_2 best respond. [**TANTRUM**] We ran TANTRUM with $n = 25$, $q_1 = 1$ and q_2 chosen randomly. These games have $> 1e12$ states;

	#	$\overline{\Delta_{\text{OPT}}}$	$\overline{\Delta_{\text{SP}}}$	$\overline{\Delta_{\text{non}}}$
RC	10	-.0247	.200	.265
TANTRUM	5	-.0262	8.89	N/A
RC+	3	N/A	N/A	.421

(a) Results for fixed parameter games



(b) EPF after 100k epochs

(c) EPF after 2M epochs

(d) Failure case

Figure 4: (a) Results for games with fixed parameters averaged over # specifies # trials. $\overline{\Delta_{\text{OPT}}}$, $\overline{\Delta_{\text{SP}}}$, and $\overline{\Delta_{\text{non}}}$ is the average difference between our method and the optimal SEFCE, subgame perfect Nash, and non-strategic leader commitment. (b)-(c) Learned EPFs at the root for RC. (d) A failure case in TANTRUM, even though learned policies are still near-optimal.

however, we can still obtain the optimal strategy due to the special structure of the game (note the subgame perfect equilibrium gives P_1 zero payoff). [**RC+**] We ran RC with $J = 9, n = 6$. Since G is large, we use approximates ($\tilde{\pi}$, $\tilde{V}$, $\underset{\sim}{V}$) obtained from $\tilde{\pi}_1^{\text{grim}}$ and $\tilde{\pi}^{\text{alt}}$. $\tilde{\pi}_1^{\text{grim}}$ is for P_2 to stay put, while $\underset{\sim}{V}$ is obtained by applying search *online* (i.e., when s appears in training) for P_2 starting from s. Thus $\tilde{\tau}(s)$ can also be computed online from $\underset{\sim}{V}$. $\tilde{\pi}^{\text{alt}}$ is obtained by running exact search to a depth of 4 (counted from the root) and then switching to a greedy algorithm. On the rare occasion that $\tilde{V}(s) < \underset{\sim}{V}(s)$, we set $\tilde{V}(s) \leftarrow \underset{\sim}{V}(s)$. We report results in Figure 4a, which show the *difference* between P_1's payoff for our method and (i) the optimal SEFCE, (ii) the subgame perfect Nash, and (iii) the non-strategic leader commitment.

Featurized TANTRUM. We allow q_1, q_2 to *vary* between stages of G, giving vectors $\mathbf{q}_i \in [1, \infty]^n$. Each trajectory uses different $\mathbf{q}_i$, which we append as features to our network, alongside the payoffs already collected for each player. For training, we draw i.i.d. samples of $\mathbf{q}_i^j \sim \exp(1) + 1$. The evaluation metric is $\kappa = R_1(\tilde{\pi})/\mathsf{OPT}$, i.e., the ratio of P_1's payoffs under $\tilde{\pi}$ compared to the optimal π. For each n, we test on 100 $\mathbf{q}$-vectors not seen during training and compare their κ against a 'greedy' strategy which recommends P_2 to accede as long as there are sufficient threats in the remainder of the game for P_1 (details in Appendix). We also stress test $\tilde{\pi}$ on a different *test* distribution $\hat{\mathbf{q}}_i^j \sim \exp(1) + 4$. We report results in Figure 5a and 5b.

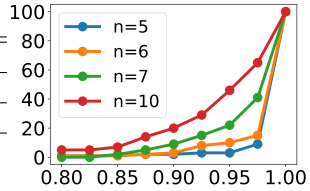
6.2 Results and Discussion

For fixed parameter games whose optimal value can be computed, we observe near optimal performance which significantly outperforms other baselines. For [RC], the average value of each an improvement of .5 is approximately equal to moving an extra half move. In [TANTRUM], the subgame perfect equilibrium is vacuous as P_1 is unable to issue threats and gets a payoff of 0. In [RC+], we are unable to fully expand the game tree, however, we still significantly outperform the non-strategic baseline.

For featurized TANTRUM, we perform near-optimally for small n, even when stress tested with out-of-distribution $\mathbf{q}$'s (Figure 5a). Performance drops as n becomes larger, which is natural as EPFs become more complex. While performance degrades as n increases, we still significantly outperform the greedy baseline. The stress test suggests that the network is not merely memorizing data.

n	$\overline{\kappa}$	grd-$\overline{\kappa}$	str-$\overline{\kappa}$
5	.993	.828	.997
6	.982	.773	.982
7	.968	.778	.921
10	.938	.775	.898

(a)



(b)

Figure 5: Results for Featurized TRANTRUM as depth n varies, based on κ, the ratio of the leader's payoff to the true optimum. (a) grd-$\overline{\kappa}$ and str-$\overline{\kappa}$ denote results for the baseline greedy method and our results when stress tested with $\mathbf{q}$ drawn from a distribution from training. (b) Proportion of trials which give $\kappa < \kappa_{\text{thresh}}$.

Figures 4b and 4c shows the learned EPFs at the root for epochs 100k and 2M, obtained directly or from one-step lookahead. As explained in Section 5, we only learn the decreasing portions of EPFs. After 2M training epochs, the predicted EPFs and one-step lookahead mirrors the true EPF in the decreasing portions, which is not the case at the beginning. At the beginning of training, many knots (red markers) are wasted on learning the 'useless' increasing portions on the left. After 2M epochs, knots (blue markers) were learning the EPF at the 'useful' decreasing regions.

Figure 4d gives an state in TANTRUM whose EPF yields high loss even after training. This failure case is not rare since TANTRUM is large. Yet, the resultant action is still optimal—in this case the promise to P_2 was $\mu_2 = -25.5$ which is precisely $\overline{V}(s)$. Like MDPs, policies can be near-optimal even with high Bellman losses in some states.

7 Conclusion

We proposed a novel method of performing FA on EPFs that allows us to efficiently solve for SEFCE. This is to the best of our knowledge, the first time a such an object has been learned from state features, leading to a FA-based method of solving Stackelberg games with performance guarantees. We hope that our approach will help to close the current gap between solving zero-sum and general-sum games.

References

Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; and Bharath, A. A. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6): 26–38.

Bakhtin, A.; Wu, D.; Lerer, A.; and Brown, N. 2021. No-Press Diplomacy from Scratch. *Advances in Neural Information Processing Systems*, 34.

Bošanskỳ, B.; Brânzei, S.; Hansen, K. A.; Lund, T. B.; and Miltersen, P. B. 2017. Computation of Stackelberg equilibria of finite sequential games. *ACM Transactions on Economics and Computation (TEAC)*, 5(4): 1–24.

Bosanský, B.; Brânzei, S.; Hansen, K. A.; Miltersen, P. B.; and Sørensen, T. B. 2015. Computation of Stackelberg Equilibria of Finite Sequential Games. *CoRR*, abs/1507.07677.

Bosansky, B.; and Cermak, J. 2015. Sequence-form algorithm for computing stackelberg equilibria in extensive-form games. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Brown, N.; and Sandholm, T. 2017. Libratus: the superhuman AI for no-limit poker. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*.

Brown, N.; and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890.

Castelletti, A.; Pianosi, F.; and Restelli, M. 2011. Multi-objective Fitted Q-Iteration: Pareto frontier approximation in one single run. In *2011 International Conference on Networking, Sensing and Control*, 260–265. IEEE.

Cermak, J.; Bosansky, B.; Durkota, K.; Lisy, V.; and Kiekintveld, C. 2016. Using correlated strategies for computing stackelberg equilibria in extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Cermák, J.; Bošanský, B.; Durkota, K.; Lisý, V.; and Kiekintveld, C. 2016. Using Correlated Strategies for Computing Stackelberg Equilibria in Extensive-form Games. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 439–445. AAAI Press.

Černỳ, J.; Bošanskỳ, B.; and Kiekintveld, C. 2018. Incremental Strategy Generation for Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, 151–168. ACM.

De Jonge, D.; and Zhang, D. 2020. Strategic negotiations for extensive-form games. *Autonomous Agents and Multi-Agent Systems*, 34(1): 1–41.

Dermed, M.; and Charles, L. 2013. *Value methods for efficiently solving stochastic games of complete and incomplete information*. Ph.D. thesis, Georgia Institute of Technology.

Dietterich, T.; and Wang, X. 2001. Batch value function approximation via support vectors. *Advances in neural information processing systems*, 14.

Fang, F.; Nguyen, T. H.; Pickles, R.; Lam, W. Y.; Clements, G. R.; An, B.; Singh, A.; Schwedock, B. C.; Tambe, M.; and Lemieux, A. 2017. PAWS-A Deployed Game-Theoretic Application to Combat Poaching. *AI Magazine*, 38(1): 23.

Foerster, J. N.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2017. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*.

Gray, J.; Lerer, A.; Bakhtin, A.; and Brown, N. 2021. Human-Level Performance in No-Press Diplomacy via Equilibrium Search. In *International Conference on Learning Representations*.

Greenwald, A.; Hall, K.; Serrano, R.; et al. 2003. Correlated Q-learning. In *ICML*, volume 3, 242–249.

Karwowski, J.; and Mańdziuk, J. 2020. Double-oracle sampling method for Stackelberg Equilibrium approximation in general-sum extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2054–2061.

Kasparov, G. 2018. Chess, a Drosophila of reasoning.

Lagoudakis, M. G.; and Parr, R. 2003. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4: 1107–1149.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*.

Letchford, J.; and Conitzer, V. 2010. Computing optimal strategies to commit to in extensive-form games. In *Proceedings of the 11th ACM conference on Electronic commerce*, 83–92. ACM.

Letchford, J.; MacDermed, L.; Conitzer, V.; Parr, R.; and Isbell, C. L. 2012. Computing optimal strategies to commit to in stochastic games. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

MacDermed, L.; Narayan, K. S.; Isbell, C. L.; and Weiss, L. 2011. Quick polytope approximation of all correlated equilibria in stochastic games. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

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. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337): 508–513.

Munos, R.; and Szepesvári, C. 2008. Finite-Time Bounds for Fitted Value Iteration. *Journal of Machine Learning Research*, 9(5).

Murray, C.; and Gordon, G. 2007. *Finding correlated equilibria in general sum stochastic games*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch.

Perolat, J.; de Vylder, B.; Hennes, D.; Tarassov, E.; Strub, F.; de Boer, V.; Muller, P.; Connor, J. T.; Burch, N.; Anthony, T.; et al. 2022. Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning. *arXiv preprint arXiv:2206.15378*.

Pérolat, J.; Strub, F.; Piot, B.; and Pietquin, O. 2017. Learning Nash equilibrium for general-sum Markov games from batch data. In *Artificial Intelligence and Statistics*, 232–241. PMLR.

Pita, J.; Jain, M.; Ordónez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; and Kraus, S. 2008. ARMOR Security for Los Angeles International Airport. In *AAAI*, 1884–1885.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schmid, M.; Moravcik, M.; Burch, N.; Kadlec, R.; Davidson, J.; Waugh, K.; Bard, N.; Timbers, F.; Lanctot, M.; Holland, Z.; et al. 2021. Player of games. *arXiv preprint arXiv:2112.03178*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Vinyals, O.; Babuschkin, I.; Chung, J.; Mathieu, M.; Jaderberg, M.; Czarnecki, W.; Dudzik, A.; Huang, A.; Georgiev, P.; Powell, R.; Ewalds, T.; Horgan, D.; Kroiss, M.; Danihelka, I.; Agapiou, J.; Oh, J.; Dalibard, V.; Choi, D.; Sifre, L.; Sulsky, Y.; Vezhnevets, S.; Molloy, J.; Cai, T.; Budden, D.; Paine, T.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Pohlen, T.; Yogatama, D.; Cohen, J.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T.; Apps, C.; Kavukcuoglu, K.; Hassabis, D.; and Silver, D. 2019. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

Wang, Y.; Shi, Z. R.; Yu, L.; Wu, Y.; Singh, R.; Joppa, L.; and Fang, F. 2019. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1401–1408.

Zhong, H.; Yang, Z.; Wang, Z.; and Jordan, M. I. 2021. Can Reinforcement Learning Find Stackelberg-Nash Equilibria in General-Sum Markov Games with Myopic Followers? *arXiv preprint arXiv:2112.13521*.

Zinkevich, M.; Greenwald, A.; and Littman, M. 2005. Cyclic equilibria in Markov games. *Advances in neural information processing systems*, 18.