A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning

Marc Lanctot DeepMind lanctot@
Vinicius Zambaldi DeepMind vzambaldi@
Audrūnas Gruslys DeepMind audrunas@
Angeliki Lazaridou DeepMind angeliki@

Karl Tuyls DeepMind karltuyls@
Julien Pérolat DeepMind perolat@
David Silver DeepMind davidsilver@
Thore Graepel DeepMind thore@

...@google.com

Abstract

To achieve general intelligence, agents must learn how to interact with others in a shared environment: this is the challenge of multiagent reinforcement learning (MARL). The simplest form is *independent reinforcement learning* (InRL), where each agent treats its experience as part of its (non-stationary) environment. In this paper, we first observe that policies learned using InRL can overfit to the other agents' policies during training, failing to sufficiently generalize during execution. We introduce a new metric, *joint-policy correlation*, to quantify this effect. We describe an algorithm for general MARL, based on approximate best responses to mixtures of policies generated using deep reinforcement learning, and empirical game-theoretic analysis to compute meta-strategies for policy selection. The algorithm generalizes previous ones such as InRL, iterated best response, double oracle, and fictitious play. Then, we present a scalable implementation which reduces the memory requirement using decoupled meta-solvers. Finally, we demonstrate the generality of the resulting policies in two partially observable settings: gridworld coordination games and poker.

1 Introduction

Deep reinforcement learning combines deep learning [58] with reinforcement learning [95, 63] to compute a policy used to drive decision-making [72, 71]. Traditionally, a single agent interacts with its environment repeatedly, iteratively improving its policy by learning from its observations. Inspired by recent success in Deep RL, we are now seeing a renewed interest in *multiagent* reinforcement learning (MARL) [91, 17, 100]. In MARL, several agents interact and learn in an environment simultaneously, either competitively such as in Go [92] and Poker [39, 106, 73], cooperatively such as when learning to communicate [23, 94, 36], or some mix of the two [59, 96, 35].

The simplest form of MARL is *independent RL* (InRL), where each learner is oblivious to the other agents and simply treats all the interaction as part of its ("localized") environment. Aside from the problem that these local environments are non-stationary and non-Markovian [56] resulting in a loss of convergence guarantees for many algorithms, the policies found can overfit to the other agents' policies and hence not generalize well. There has been relatively little work done in RL community on overfitting to the environment [103, 68], but we argue that this is particularly important in multiagent settings where one must react dynamically based on the observed behavior of others. Classical techniques collect or approximate extra information such as the joint values [61, 19, 29, 55],

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

use adaptive learning rates [12], adjust the frequencies of updates [48, 80], or dynamically respond to the other agents actions online [62, 50]. However, with the notable exceptions of very recent work [22, 79], they have focused on (repeated) matrix games and/or the fully-observable case.

There have been several proposals for treating partial observability in the multiagent setting. When the model is fully known and the setting is strictly adversarial with two players, there are policy iteration methods based on regret minimization that scale very well when using domain-specific abstractions [27, 14, 46, 47], which was a major component of the expert no-limit poker AI Libratus [15]; recently these methods were combined with deep learning to create an expert no-limit poker AI called DeepStack [73]. There is a significant amount of work that deals with the case of decentralized cooperative problems [75, 78], and in the general setting by extending the notion of belief states and Bayesian updating from POMDPs [28]. These models are quite expressive, and the resulting algorithms are fairly complex. In practice, researchers often resort to approximate forms, by sampling or exploiting structure, to ensure good performance due to intractability [41, 2, 67].

In this paper, we introduce a new metric for quantifying the correlation effects of policies learned by independent learners, and demonstrate the severity of the overfitting problem. These coordination problems have been well-studied in the fully-observable cooperative case [69]: we observe similar problems in a partially-observed mixed cooperative/competitive setting and, and we show that the severity increases as the environment becomes more partially-observed. We propose a new algorithm based on economic reasoning [81], which uses (i) deep reinforcement learning to compute best responses to a distribution over policies, and (ii) empirical game-theoretic analysis to compute new meta-strategy distributions. As is common in the MARL setting, we assume centralized training for decentralized execution: policies are represented as separate neural networks and there is no sharing of gradients nor architectures among agents. The basic form uses a centralized payoff table, which is removed in the distributed, decentralized form that requires less space.

2 Background and Related Work

In this section, we start with basic building blocks necessary to describe the algorithm. We interleave this with the most relevant previous work for our setting. Several components of the general idea have been (re)discovered many times across different research communities, each with slightly different but similar motivations. One aim here is therefore to unify the algorithms and terminology.

A **normal-form game** is a tuple (Π, U, n) where n is the number of players, $\Pi = (\Pi_1, \cdots, \Pi_n)$ is the set of policies (or strategies, one for each player $i \in [[n]]$, where $[[n]] = \{1, \cdots, n\}$), and $U : \Pi \rightarrow \Re^n$ is a payoff table of utilities for each joint policy played by all players. **Extensive-form games** extend these formalisms to the multistep sequential case (*e.g.* poker).

Players try to maximize their own expected utility. Each player does this by choosing a policy from Π_i, or by sampling from a mixture (distribution) over them $\sigma_i \in \Delta(\Pi_i)$. In this multiagent setting, the quality of σ_i depends on other players' strategies, and so it cannot be found nor assessed independently. Every finite extensive-form game has an equivalent normal-form [53], but since it is exponentially larger, most algorithms have to be adapted to handle the sequential setting directly.

There are several algorithms for computing strategies. In zero-sum games (where $\forall \pi \in \Pi, \vec{1} \cdot U(\pi) = 0$), one can use e.g. linear programming, fictitious play [13], replicator dynamics [98], or regret minimization [8]. Some of these techniques have been extended to extensive (sequential) form [39, 25, 54, 108] with an exponential increase in the size of the state space. However, these extensions have almost exclusively treated the two-player case, with some notable exceptions [54, 26]. Fictitious play also converges in potential games which includes cooperative (identical payoff) games.

The **double oracle** (DO) algorithm [70] solves a set of (two-player, normal-form) subgames induced by subsets $\Pi^t \subset \Pi$ at time t. A payoff matrix for the subgame G_t includes only those entries corresponding to the strategies in Π^t. At each time step t, an equilibrium $\sigma^{*,t}$ is obtained for G^t, and to obtain G^{t+1} each player adds a best response $\pi_i^{t+1} \in \text{BR}(\sigma_{-i}^{*,t})$ from the full space Π_i, so for all i, $\Pi_i^{t+1} = \Pi_i^t \cup \{\pi_i^{t+1}\}$. The algorithm is illustrated in Figure 1. Note that finding an equilibrium in a zero-sum game takes time polynomial in $|\Pi^t|$, and is PPAD-complete for general-sum [90].

Clearly, DO is guaranteed to converge to an equilibrium in two-player games. But, in the worst-case, the entire strategy space may have to be enumerated. For example, this is necessary for Rock-Paper-

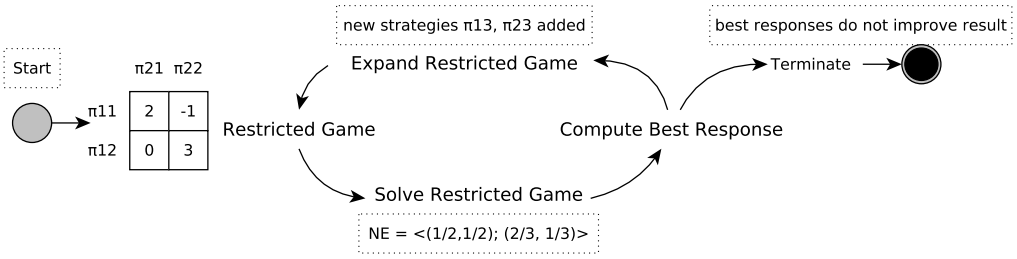


Figure 1: The Double Oracle Algorithm. Figure taken from [10] with authors' permission.

Scissors, whose only equilibrium has full support $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. However, there is evidence that support sizes shrink for many games as a function of episode length, how much hidden information is revealed and/or effects it has on the payoff [64, 86, 10]. Extensions to the extensive-form games have been developed [66, 9, 10] but still large state spaces are problematic due to the curse of dimensionality.

Empirical game-theoretic analysis (EGTA) is the study of meta-strategies obtained through simulation in complex games [101, 102]. An **empirical game**, much smaller in size than the full game, is constructed by discovering strategies, and meta-reasoning about the strategies to navigate the strategy space. This is necessary when it is prohibitively expensive to explicitly enumerate the game's strategies. Expected utilities for each joint strategy are estimated and recorded in an **empirical payoff table**. The empirical game is analyzed, and the simulation process continues. EGTA has been employed in trading agent competitions (TAC) and automated bidding auctions.

One study used evolutionary dynamics in the space of known expert meta-strategies in Poker [82]. Recently, reinforcement learning has been used to *validate* strategies found via EGTA [105]. In this work, we aim to discover new strategies through learning. However, instead of computing exact best responses, we compute approximate best responses using reinforcement learning. A few epochs of this was demonstrated in continuous double auctions using tile coding [87]. This work follows up in this line, running more epochs, using modern function approximators (deep networks), a scalable implementation, and with a focus on finding policies that can generalize across contexts.

A key development in recent years is deep learning [58]. While most work in deep learning has focused on supervised learning, impressive results have recently been shown using deep neural networks for reinforcement learning, e.g. [92, 38, 72, 76]. For instance, Mnih et al. [72] train policies for playing Atari video games and 3D navigation [71], given only screenshots. Silver et al. introduced AlphaGo [92, 93], combining deep RL with Monte Carlo tree search, outperforming human experts.

Computing approximate responses is more computationally feasible, and fictitious play can handle approximations [42, 60]. It is also more biologically plausible given natural constraints of bounded rationality. In **behavioral game theory** [104, 83], the focus is to *predict* actions taken by humans, and the responses are intentionally constrained to increase predictive ability. A recent work uses a deep learning architecture [34]. The work that closely resembles ours is **level-k** thinking [20] where level k agents respond to level $k-1$ agents, and more closely **cognitive hierarchy** [18], in which responses are to distributions over levels $\{0, 1, \ldots, k-1\}$. However, our goals and motivations are very different: we use the setup as a means to produce more general policies rather than to predict human behavior. Furthermore, we consider the sequential setting rather than normal-form games.

Lastly, there has been several studies from the literature on co-evolutionary algorithms; specifically, how learning cycles and overfitting to the current populations can be mitigated [77, 85, 52].

3 Policy-Space Response Oracles

We now present our main conceptual algorithm, policy-space response oracles (PSRO). The algorithm is a natural generalization of Double Oracle where the meta-game's choices are policies rather than actions. It also generalizes Fictitious Self-Play [39, 40]. Unlike previous work, any meta-solver can be plugged in to compute a new meta-strategy. In practice, parameterized policies (function approximators) are used to generalize across the state space without requiring any domain knowledge.

The process is summarized in Algorithm 1. The meta-game is represented as an empirical game, starting with a single policy (uniform random) and growing, each epoch, by adding policies ("oracles")

Algorithm 1: Policy-Space Response Oracles

input : initial policy sets for all players Π
Compute exp. utilities U^{Π} for each joint $\pi \in \Pi$
Initialize meta-strategies $\sigma_i = \text{UNIFORM}(\Pi_i)$
while *epoch* e *in* $\{1, 2, \cdots\}$ **do**
  for *player* $i \in [[n]]$ **do**
    for *many episodes* **do**
      Sample $\pi_{-i} \sim \sigma_{-i}$
      Train oracle π_i' over $\rho \sim (\pi_i', \pi_{-i})$
    $\Pi_i = \Pi_i \cup \{\pi_i'\}$
  Compute missing entries in U^{Π} from Π
  Compute a meta-strategy σ from U^{Π}
Output current solution strategy σ_i for player i

Algorithm 2: Deep Cognitive Hierarchies

input : player number i, level k
while *not terminated* **do**
  CHECKLOADMS($\{j | j \in [[n]], j \neq i\}, k$)
  CHECKLOADORACLES($j \in [[n]], k' \leq k$)
  CHECKSAVEMS($\sigma_{i,k}$)
  CHECKSAVEORACLE($\pi_{i,k}$)
  Sample $\pi_{-i} \sim \sigma_{-i,k}$
  Train oracle $\pi_{i,k}$ over $\rho_1 \sim (\pi_{i,k}, \pi_{-i})$
  if *iteration number* $\bmod T_{ms} = 0$ **then**
    Sample $\pi_i \sim \sigma_{i,k}$
    Compute $u_i(\rho_2)$, where $\rho_2 \sim (\pi_i, \pi_{-i})$
    Update stats for π_i and update $\sigma_{i,k}$
Output $\sigma_{i,k}$ for player i at level k

that approximate best responses to the meta-strategy of the other players. In (episodic) partially observable multiagent environments, when the other players are fixed the environment becomes Markovian and computing a best response reduces to solving a form of MDP [30]. Thus, any reinforcement learning algorithm can be used. We use deep neural networks due to the recent success in reinforcement learning. In each episode, one player is set to *oracle*(learning) mode to train π_i', and a fixed policy is sampled from the opponents' meta-strategies ($\pi_{-i} \sim \sigma_{-i}$). At the end of the epoch, the new oracles are added to their policy sets Π_i, expected utilities for new policy combinations are computed via simulation and added to the empirical tensor U^{Π}, which takes time exponential in $|\Pi|$.

Define $\Pi^T = \Pi^{T-1} \cup \{\pi'\}$ as the policy space including the currently learning oracles, and $|\sigma_i| = |\Pi_i^T|$ for all $i \in [[n]]$. Iterated best response is an instance of PSRO with $\sigma_{-i} = (0, 0, \cdots, 1, 0)$. Similarly, Independent RL and fictitious play are instances of PSRO with $\sigma_{-i} = (0, 0, \cdots, 0, 1)$ and $\sigma_{-i} = (1/K, 1/K, \cdots, 1/K, 0)$, respectively, where $K = |\Pi_{-i}^{T-1}|$. Double Oracle is an instance of PSRO with $n = 2$ and σ^T set to a Nash equilibrium profile of the meta-game $(\Pi^{T-1}, U^{\Pi^{T-1}})$.

An exciting question is what can happen with (non-fixed) meta-solvers outside this known space? Fictitious play is agnostic to the policies it is responding to; hence it can only sharpen the meta-strategy distribution by repeatedly generating the same best responses. On the other hand, responses to equilibrium strategies computed by Double Oracle will (i) overfit to a specific equilibrium in the n-player or general-sum case, and (ii) be unable to generalize to parts of the space not reached by any equilibrium strategy in the zero-sum case. Both of these are undesirable when computing general policies that should work well in any context. We try to balance these problems of overfitting with a compromise: meta-strategies with full support that force (mix in) γ exploration over policy selection.

3.1 Meta-Strategy Solvers

A meta-strategy solver takes as input the empirical game (Π, U^{Π}) and produces a meta-strategy σ_i for each player i. We try three different solvers: regret-matching, Hedge, and projected replicator dynamics. These specific meta-solvers accumulate values for each policy ("arm") and an aggregate value based on all players' meta-strategies. We refer to $u_i(\sigma)$ as player i's expected value given all players' meta-strategies and the current empirical payoff tensor U^{Π} (computed via multiple tensor dot products.) Similarly, denote $u_i(\pi_{i,k}, \sigma_{-i})$ as the expected utility if player i plays their $k^{th} \in [[K]] \cup \{0\}$ policy and the other players play with their meta-strategy σ_{-i}. Our strategies use an exploration parameter γ, leading to a lower bound of $\frac{\gamma}{K+1}$ on the probability of selecting any $\pi_{i,k}$.

The first two meta-solvers (Regret Matching and Hedge) are straight-forward applications of previous algorithms, so we defer the details to Appendix A. Here, we introduce a new solver we call **projected replicator dynamics** (PRD). From Appendix A, when using the asymmetric replicator dynamics, *e.g.* with two players, where $U^{\Pi} = (\mathbf{A}, \mathbf{B})$, the change in probabilities for the k^{th} component (*i.e.,* the policy $\pi_{i,k}$) of meta-strategies $(\sigma_1, \sigma_2) = (\mathbf{x}, \mathbf{y})$ are:

$$\frac{dx_k}{dt} = x_k[(\mathbf{A}\mathbf{y})_k - \mathbf{x}^T\mathbf{A}\mathbf{y}], \qquad \frac{dy_k}{dt} = y_k[(\mathbf{x}^T\mathbf{B})_k - \mathbf{x}^T\mathbf{B}\mathbf{y}],$$

To simulate the replicator dynamics in practice, discretized updates are simulated using a step-size of δ. We add a projection operator $P(\cdot)$ to these equations that guarantees exploration: $\mathbf{x} \leftarrow P(\mathbf{x} + \delta \frac{d\mathbf{x}}{dt})$, $\mathbf{y} \leftarrow P(\mathbf{y} + \delta \frac{d\mathbf{y}}{dt})$, where $P(\mathbf{x}) = \text{argmin}_{\mathbf{x}' \in \Delta_\gamma^{K+1}}\{||\mathbf{x}' - \mathbf{x}||\}$, if any $x_k < \gamma/(K+1)$ or $\mathbf{x}$ otherwise, and $\Delta_\gamma^{K+1} = \{\mathbf{x} \mid x_k \geq \frac{\gamma}{K+1}, \sum_k x_k = 1\}$ is the γ-exploratory simplex of size $K+1$. This enforces exploratory $\sigma_i(\pi_{i,k}) \geq \gamma/(K+1)$. The PRD approach can be understood as directing exploration in comparison to standard replicator dynamics approaches that contain isotropic diffusion or mutation terms (which assume undirected and unbiased evolution), for more details see [99].

3.2 Deep Cognitive Hierarchies

While the generality of PSRO is clear and appealing, the RL step can take a long time to converge to a good response. In complex environments, much of the basic behavior that was learned in one epoch may need to be relearned when starting again from scratch; also, it may be desirable to run many epochs to get oracle policies that can recursively reason through deeper levels of contingencies.

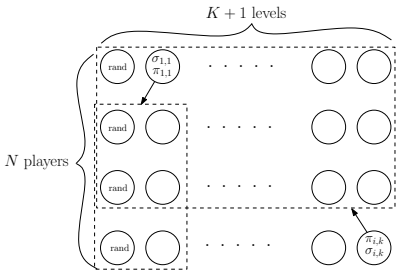


Figure 2: Overview of DCH

To overcome these problems, we introduce a practical parallel form of PSRO. Instead of an unbounded number of epochs, we choose a fixed number of *levels* in advance. Then, for an n-player game, we start nK processes in parallel (level 0 agents are uniform random): each one trains a single oracle policy $\pi_{i,k}$ for player i and level k and updates its own meta-strategy $\sigma_{i,k}$, saving each to a central disk periodically. Each process also maintains copies of all the other oracle policies $\pi_{j,k' \leq k}$ at the current and lower levels, as well as the meta-strategies at the current level $\sigma_{-i,k}$, which are periodically refreshed from a central disk. We circumvent storing U^Π explicitly by updating the meta-strategies online. We call this a Deep Cognitive Hierarchy (DCH), in reference to Camerer, Ho, & Chong's model augmented with deep RL. Example oracle response dynamics are shown in Figure 2, and pseudo-code in Algorithm 2.

Since each process uses slightly out-dated copies of the other process's policies and meta-strategies, DCH approximates PSRO. Specifically, it trades away accuracy of the correspondence to PSRO for practical efficiency and, in particular, scalability. Another benefit of DCH is an asymptotic reduction in total space complexity. In PSRO, for K policies and n players, the space required to store the empirical payoff tensor is K^n. Each process in DCH stores nK policies of fixed size, and n meta-strategies (and other tables) of size bounded by $k \leq K$. Therefore the total space required is $O(nK \cdot (nK + nK)) = O(n^2K^2)$. This is possible is due to the use of *decoupled* meta-solvers, which compute strategies online without requiring a payoff tensor U^Π, which we describe now.

3.2.1 Decoupled Meta-Strategy Solvers

In the field of online learning, the experts algorithms ("full information" case) receive information about each arm at every round. In the bandit ("partial information") case, feedback is only given for the arm that was pulled. Decoupled meta-solvers are essentially sample-based adversarial bandits [16] applied to games. Empirical strategies are known to converge to Nash equilibria in certain classes of games (i.e. zero-sum, potential games) due to the folk theorem [8].

We try three: decoupled regret-matching [33], Exp3 (decoupled Hedge) [3], and decoupled PRD. Here again, we use exploratory strategies with γ of the uniform strategy mixed in, which is also necessary to ensure that the estimates are unbiased. For decoupled PRD, we maintain running averages for the overall average value an value of each arm (policy). Unlike in PSRO, in the case of DCH, one sample is obtained at a time and the meta-strategy is updated periodically from online estimates.

4 Experiments

In all of our experiments, oracles use Reactor [31] for learning, which has achieved state-of-the-art results in Atari game-playing. Reactor uses Retrace(λ) [74] for off-policy policy evaluation, and β-Leave-One-Out policy gradient for policy updates, and supports recurrent network training, which could be important in trying to match online experiences to those observed during training.

The action spaces for each player are identical, but the algorithms do not require this. Our implementation differs slightly from the conceptual descriptions in Section 3; see App. C for details.

First-Person Gridworld Games. Each agent has a local field-of-view (making the world partially observable), sees 17 spaces in front, 10 to either side, and 2 spaces behind. Consequently, observations are encoded as 21x20x3 RGB tensors with values 0 – 255. Each agent has a choice of turning left or right, moving forward or backward, stepping left or right, not moving, or casting an endless light beam in their current direction. In addition, the agent has two composed actions of moving forward and turning. Actions are executed simultaneously, and order of resolution is randomized. Agents start on a random spawn point at the beginning of each episode. If an agent is touched ("tagged") by another agent's light beam twice, then the target agent is immediately teleported to a spawn point. In *laser tag*, the source agent then receives 1 point of reward for the tag. In another variant, *gathering*, there is no tagging but agents can collect apples, for 1 point per apple, which refresh at a fixed rate. In *pathfind*, there is no tagging nor apples, and both agents get 1 point reward when both reach their destinations, ending the episode. In every variant, an episode consists of 1000 steps of simulation. Other details, such as specific maps, can be found in Appendix D.

Leduc Poker is a common benchmark in Poker AI, consisting of a six-card deck: two suits with three cards (Jack, Queen, King) each. Each player antes 1 chip to play, and receives one private card. There are two rounds of betting, with a maximum of two raises each, whose values are 2 and 4 chips respectively. After the first round of betting, a single public card is revealed. The input is represented as in [40], which includes one-hot encodings of the private card, public card, and history of actions. Note that we use a more difficult version than in previous work; see Appendix D.1 for details.

4.1 Joint Policy Correlation in Independent Reinforcement Learning

To identify the effect of overfitting in independent reinforcement learners, we introduce **joint policy correlation** (JPC) matrices. To simplify the presentation, we describe here the special case of symmetric two-player games with non-negative rewards; for a general description, see Appendix B.2.

Values are obtained by running D instances of the same experiment, differing only in the seed used to initialize the random number generators. Each experiment $d \in [[D]]$ (after many training episodes) produces policies (π_1^d, π_2^d). The entries of each $D \times D$ matrix shows the mean return over $T = 100$ episodes, $\sum_{t=1}^{T} \frac{1}{T}(R_1^t + R_2^t)$, obtained when player 1 uses row policy $\pi_1^{d_i}$ and and player 2 uses column policy $\pi_2^{d_j}$. Hence, entries on the diagonals represent returns for policies that learned together (*i.e.,* same instance), while off-diagonals show returns from policies that trained in separate instances.

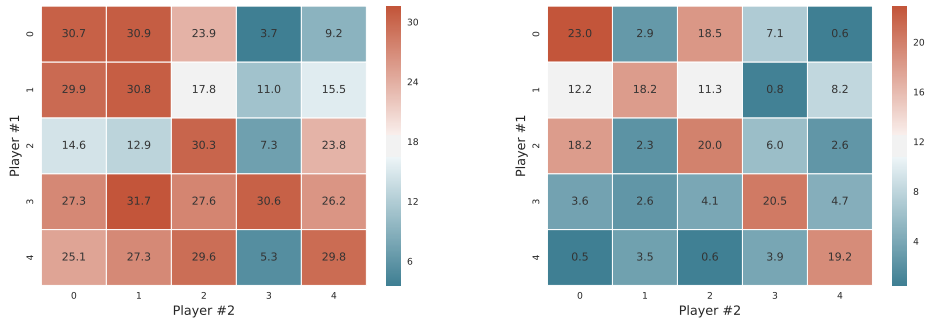


Figure 3: Example JPC matrices for InRL on Laser Tag small2 map (left) and small4 (right).

From a JPC matrix, we compute an **average proportional loss** in reward as $R_{-} = (\bar{D} - \bar{O})/\bar{D}$ where $\bar{D}$ is the mean value of the diagonals and $\bar{O}$ is the mean value of the off-diagonals. E.g. in Figure 3: $D = 30.44, O = 20.03, R_{-} = 0.342$. Even in a simple domain with almost full observability (small2), an independently-learned policy could expect to lose 34.2% of its reward when playing with another independently-learned policy even though it *was trained under identical circumstances*! This clearly demonstrates an important problem with independent learners. In the other variants (gathering and pathfind), we observe no JPC problem, presumably because coordination is not required and the policies are independent. Results are summarized in Table 1. We have also noticed similar effects when using DQN [72] as the oracle training algorithm; see Appendix B.1 for example videos.

Environment	Map	InRL			DCH(Reactor, 2, 10)			JPC Reduction
		$\bar{D}$	$\bar{O}$	R_-	$\bar{D}$	$\bar{O}$	R_-	
Laser Tag	small2	30.44	20.03	0.342	28.20	26.63	0.055	28.7 %
Laser Tag	small3	23.06	9.06	0.625	27.00	23.45	0.082	54.3 %
Laser Tag	small4	20.15	5.71	0.717	18.72	15.90	0.150	56.7 %
Gathering	field	147.34	146.89	0.003	139.70	138.74	0.007	–
Pathfind	merge	108.73	106.32	0.022	90.15	91.492	< 0	–

Table 1: Summary of JPC results in first-person gridworld games.

We see that a (level 10) DCH agent reduces the JPC problem significantly. On small2, DCH reduces the expected loss down to 5.5%, 28.7% lower than independent learners. The problem gets larger as the map size grows and problem becomes more partially observed, up to a severe 71.7% average loss. The reduction achieved by DCH also grows from 28.7% to 56.7%.

Is the Meta-Strategy Necessary During Execution? The figures above represent the fully-mixed strategy $\sigma_{i,10}$. We also analyze JPC for only the highest-level policy $\pi_{i,10}$ in the laser tag levels. The values are larger here: $R_- = 0.147, 0.27, 0.118$ for small2-4 respectively, showing the importance of the meta-strategy. However, these are still significant reductions in JPC: 19.5%, 36.5%, 59.9%.

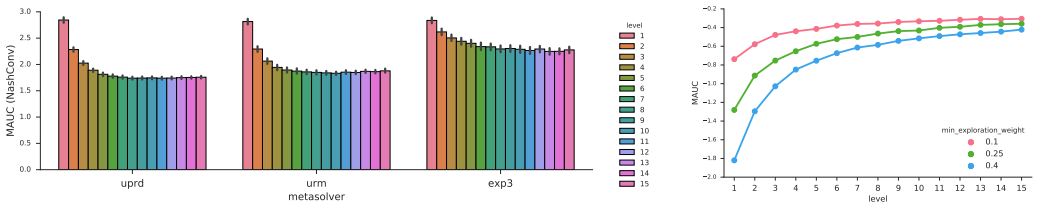
How Many Levels? On small4, we also compute values for level 5 and level 3: $R_- = 0.156$ and $R_- = 0.246$, corresponding to reductions in JPC of 56.1% and 44%, respectively. Level 5 reduces JPC by a similar amount as level 10 (56.1% vs 56.7%), while level 3 less so (44% vs. 56.1%.)

4.2 Learning to Safely Exploit and Indirectly Model Opponents in Leduc Poker

We now show results for a Leduc poker where strong benchmark algorithms exist, such as counterfactual regret (CFR) minimization [108, 11]. We evaluate our policies using two metrics: the first is performance against fixed players (random, CFR's average strategy after 500 iterations "cfr500", and a purified version of "cfr500pure" that chooses the action with highest probability.) The second is commonly used in poker AI: $\text{NASHCONV}(\sigma) = \sum_i^n \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}) - u_i(\sigma)$, representing, in total, how much each player gains by deviating to their best response (unilaterally), a value that can be interpreted as a distance from a Nash equilibrium (called **exploitability** in the two-player setting). NashConv is easy to compute in small enough games [45]; for CFR's values see Appendix E.2.

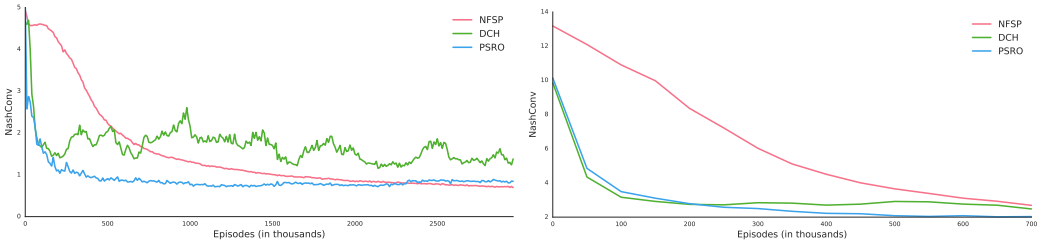
Effect of Exploration and Meta-Strategy Overview. We now analyze the effect of the various meta-strategies and exploration parameters. In Figure 4, we measure the mean area-under-the-curve (MAUC) of the NashConv values for the last (right-most) 32 values in the NashConv graph, and exploration rate of $\gamma = 0.4$. Figures for the other values of γ are in Appendix E, but we found this value of γ works best for minimizing NashConv. Also, we found that decoupled replicator dynamics works best, followed by decoupled regret-matching and Exp3. Also, it seems that the higher the level, the lower the resulting NashConv value is, with diminishing improvements. For exploitation, we found that $\gamma = 0.1$ was best, but the meta-solvers seemed to have little effect (see Figure 11.)

Comparison to Neural Fictitious Self-Play. We now compare to Neural Fictitious Self-Play (NFSP) [40], an implementation of fictitious play in sequential games using reinforcement learning. Note that NFSP, PSRO, and DCH are all sample-based learning algorithms that use general function approximation, whereas CFR is a tabular method that requires a full game-tree pass per iteration. NashConv graphs are shown for {2,3}-player in Figure 5, and performance vs. fixed bots in Figure 6.



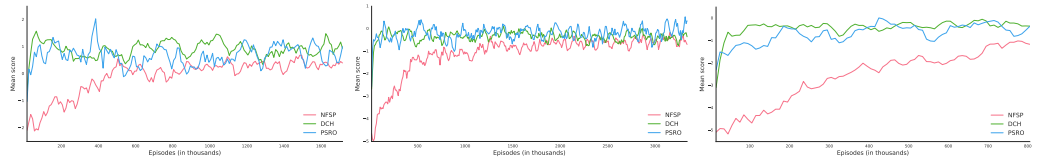
(a) (b)

Figure 4: (a) Effect of DCH parameters on NashConv in 2 player Leduc Poker. Left: decoupled PRD, Middle: decoupled RM, Right: Exp3, and (b) MAUC of the exploitation graph against cfr500.



(a) 2 players (b) 3 players

Figure 5: Exploitability for NFSP x DCH x PSRO.



(a) Random bots as ref. set (b) 2-player CFR500 bots as ref. set (c) 3-player CFR500 bots as ref. set

Figure 6: Evaluation against fixed set of bots. Each data point is an average of the four latest values.

We observe that DCH (and PSRO) converge faster than NFSP at the start of training, possibly due to a better meta-strategy than the uniform random one used in fictitious play. The convergence curves eventually plateau: DCH in two-player is most affected, possibly due to the asynchronous nature of the updates, and NFSP converges to a lower exploitability in later episodes. We believe that this is due to NFSP's ability to learn a more accurate mixed average strategy at states far down in the tree, which is particularly important in poker, whereas DCH and PSRO mix at the top over full policies.

On the other hand, we see that PSRO/DCH are able to achieve higher performance against the fixed players. Presumably, this is because the policies produced by PSRO/DCH are better able to recognize flaws in the weaker opponent's policies, since the oracles are specifically trained for this, and dynamically adapt to the exploitative response during the episode. So, NFSP is computing a safe equilibrium while PSRO/DCH may be trading convergence precision for the ability to adapt to a range of different play observed during training, in this context computing a robust counter-strategy [44, 24].

5 Conclusion and Future Work

In this paper, we quantify a severe problem with independent reinforcement learners, joint policy correlation (JPC), that limits the generality of these approaches. We describe a generalized algorithm for multiagent reinforcement learning that subsumes several previous algorithms. In our experiments, we show that PSRO/DCH produces general policies that significantly reduce JPC in partially-observable coordination games, and robust counter-strategies that safely exploit opponents in a common competitive imperfect information game. The generality offered by PSRO/DCH can be seen as a form of "opponent/teammate regularization", and has also been observed recently in practice [65, 5]. We emphasize the game-theoretic foundations of these techniques, which we hope will inspire further investigation into algorithm development for multiagent reinforcement learning.

In future work, we will consider maintaining diversity among oracles via loss penalties based on policy dissimilarity, general response graph topologies, environments such as emergent language games [57] and RTS games [97, 84], and other architectures for prediction of behavior, such as opponent modeling [37] and imagining future states via auxiliary tasks [43]. We would also like to investigate fast online adaptation [1, 21] and the relationship to computational Theory of Mind [107, 4], as well as generalized (transferable) oracles over similar opponent policies using successor features [6].

Acknowledgments. We would like to thank DeepMind and Google for providing an excellent research environment that made this work possible. Also, we would like to thank the anonymous reviewers and several people for helpful comments: Johannes Heinrich, Guy Lever, Remi Munos, Joel Z. Leibo, Janusz Marecki, Tom Schaul, Noam Brown, Kevin Waugh, Georg Ostrovski, Sriram Srinivasan, Neil Rabinowitz, and Vicky Holgate.

References

[1] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *CoRR*, abs/1710.03641, 2017.

[2] Christopher Amato and Frans A. Oliehoek. Scalable planning and learning for multiagent POMDPs. In *AAAI15*, pages 1995–2002, January 2015.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[4] C.L. Baker, R.R. Saxe, and J.B. Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the Thirty-Third Annual Conference of the Cognitive Science Society*, pages 2469–2474, 2011.

[5] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *CoRR*, abs/1710.03748, 2017.

[6] André Barreto, Will Dabney, Rémi Munos, Jonathan Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Transfer in reinforcement learning with successor features and generalised policy improvement. In *Proceedings of the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS 2017)*, 2017. To appear. Preprint available at `http://arxiv.org/abs/1606.05312`.

[7] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *J. Artif. Intell. Res. (JAIR)*, 53:659–697, 2015.

[8] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*, chapter 4. Cambridge University Press, 2007.

[9] Branislav Bosansky, Viliam Lisy, Jiri Cermak, Roman Vitek, and Michal Pechoucek. Using double-oracle method and serialized alpha-beta search for pruning in simultaneous moves games. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[10] Branislav Bošanský, Viliam Lisý, Marc Lanctot, Jiří Čermák, and Mark H.M. Winands. Algorithms for computing strategies in two-player simultaneous move games. *Artificial Intelligence*, 237:1–40, 2016.

[11] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up Limit Hold'em Poker is solved. *Science*, 347(6218):145–149, January 2015.

[12] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[13] G. W. Brown. Iterative solutions of games by fictitious play. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. John Wiley & Sons, Inc., 1951.

[14] Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas Hold'em agent. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 7–15. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[15] Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. *CoRR*, abs/1705.02955, 2017.

[16] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[17] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[18] Colin F. Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 2004.

[19] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, 1998.

[20] M. A. Costa-Gomes and V. P. Crawford. Cognition and behavior in two-person guessing games: An experimental study. *The American Economy Review*, 96(6):1737–1768, 2006.

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[22] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, 2017.

[23] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[24] Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. *ACM Transactions on Economics and Computation (TEAC)*, 3(2):1–28, 2015. Special issue on selected papers from EC-12.

[25] N. Gatti, F. Panozzo, and M. Restelli. Efficient evolutionary dynamics with extensive-form games. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 335–341, 2013.

[26] Richard Gibson. Regret minimization in non-zero-sum games with applications to building champion multiplayer computer poker agents. *CoRR*, abs/1305.0034, 2013.

[27] A. Gilpin. *Algorithms for Abstracting and Solving Imperfect Information Games*. PhD thesis, Carnegie Mellon University, 2009.

[28] Gmytrasiewicz and Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[29] Amy Greenwald and Keith Hall. Correlated Q-learning. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 242–249, 2003.

[30] Amy Greenwald, Jiacui Li, and Eric Sodomka. Solving for best responses and equilibria in extensive-form games with reinforcement learning methods. In *Rohit Parikh on Logic, Language and Society*, volume 11 of *Outstanding Contributions to Logic*, pages 185–226. Springer International Publishing, 2017.

[31] Audrunas Gruslys, Mohammad Gheshlaghi Azar, Marc G. Bellemare, and Remi Munos. The Reactor: A sample-efficient actor-critic architecture. *CoRR*, abs/1704.04651, 2017.

[32] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[33] Sergiu Hart and Andreu Mas-Colell. A reinforcement procedure leading to correlated equilibrium. In *Economics Essays: A Festschrift for Werner Hildenbrand*. Springer Berlin Heidelberg, 2001.

[34] Jason S. Hartford, James R. Wright, and Kevin Leyton-Brown. Deep learning for predicting human strategic behavior. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[35] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2016.

[36] Matthew John Hausknecht. *Cooperation and communication in multiagent deep reinforcement learning*. PhD thesis, University of Texas at Austin, Austin, USA, 2016.

[37] He He, Jordan Boyd-Graber, Kevin Kwok, , and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *In Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 1804–1813, 2016.

[38] Nicolas Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2944–2952, 2015.

[39] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, 2015.

[40] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016.

[41] Trong Nghia Hoang and Kian Hsiang Low. Interactive POMDP lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2298–2305. AAAI Press, 2013.

[42] Josef Hofbauer and William H. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294, 11 2002.

[43] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016.

[44] M. Johanson, M. Zinkevich, and M. Bowling. Computing robust counter-strategies. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1128–1135, 2008. A longer version is available as a University of Alberta Technical Report, TR07-15.

[45] Michael Johanson, Michael Bowling, Kevin Waugh, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 258–265, 2011.

[46] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[47] Michael Bradley Johanson. *Robust Strategies and Counter-Strategies: From Superhuman to Optimal Play*. PhD thesis, University of Alberta, 2016. `http://johanson.ca/publications/theses/2016-johanson-phd-thesis/2016-johanson-phd-thesis.pdf`.

[48] Michael Kaisers and Karl Tuyls. Frequency adjusted multi-agent Q-learning. In *9th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, pages 309–316, 2010.

[49] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] M. Kleiman-Weiner, M. K. Ho, J. L. Austerweil, M. L. Littman, and J. B. Tenenbaum. Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*, 2016.

[51] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC '94)*, pages 750–759, 1994.

[52] Kostas Kouvaris, Jeff Clune, Loizos Kounios, Markus Brede, and Richard A. Watson. How evolution learns to generalise: Using the principles of learning theory to understand the evolution of developmental organisation. *PLOS Computational Biology*, 13(4):1–20, 04 2017.

[53] H. W. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, 2:193–216, 1953.

[54] Marc Lanctot. Further developments of extensive-form replicator dynamics using the sequence-form representation. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1257–1264, 2014.

[55] M. Lauer and M. Riedmiller. Reinforcement learning for stochastic cooperative multi-agent systems. In *Proceedings of the AAMAS '04, New York*, 2004.

[56] Guillaume J. Laurent, Laëtitia Matignon, and Nadine Le Fort-Piat. The world of independent learners is not Markovian. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 15(1):55–64, March 2011.

[57] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2017.

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[59] Joel Z. Leibo, Vinicius Zambaldia, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the Sixteenth International Conference on Autonomous Agents and Multiagent Systems*, 2017.

[60] David S. Leslie and Edmund J. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

[61] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

[62] Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 322–328, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[63] Michael L. Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521:445–451, 2015.

[64] J. Long, N. R. Sturtevant, M. Buro, and T. Furtak. Understanding the success of perfect information Monte Carlo sampling in game tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 134–140, 2010.

[65] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275, 2017.

[66] N. Burch M. Zinkevich, M. Bowling. A new algorithm for generating equilibria in massive zero-sum games. In *Proceedings of the Twenty-Seventh Conference on Artificial Intelligence (AAAI-07)*, 2007.

[67] Janusz Marecki, Tapana Gupta, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Not all agents are equal: Scaling up distributed pomdps for agent networks. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2008.

[68] Vukosi N. Marivate. *Improved Empirical Methods in Reinforcement Learning Evaluation*. PhD thesis, Rutgers, New Brunswick, New Jersey, 2015.

[69] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(01):1–31, 2012.

[70] H.B. McMahan, G. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[71] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

[72] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[73] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 358(6362), October 2017.

[74] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

[75] Ranjit Nair. *Coordinating multiagent teams in uncertain domains using distributed POMDPs*. PhD thesis, University of Southern California, Los Angeles, USA, 2004.

[76] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder P. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2863–2871, 2015.

[77] F.A. Oliehoek, E.D. de Jong, and N. Vlassis. The parallel Nash memory for asymmetric games. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2006.

[78] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer-Briefs in Intelligent Systems. Springer, 2016. Authors' pre-print.

[79] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, 2017.

[80] Liviu Panait, Karl Tuyls, and Sean Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *Journal of Machine Learning Research*, 9:423–457, 2008.

[81] David C. Parkes and Michael P. Wellman. Economic reasoning and artificial intelligence. *Science*, 349(6245):267–272, 2015.

[82] Marc Ponsen, Karl Tuyls, Michael Kaisers, and Jan Ramon. An evolutionary game theoretic analysis of poker strategies. *Entertainment Computing*, 2009.

[83] James R.Wright and Kevin Leyton-Brown. Predicting human behavior in unrepeated, simultaneous-move games. *Games and Economic Behavior*, 106:16–37, November 2017.

[84] F. Sailer, M. Buro, and M. Lanctot. Adversarial planning through strategy simulation. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 37–45, 2007.

[85] Spyridon Samothrakis, Simon Lucas, Thomas Philip Runarsson, and David Robles. Coevolving Game-Playing Agents: Measuring Performance and Intransitivities. *IEEE Transactions on Evolutionary Computation*, April 2013.

[86] Martin Schmid, Matej Moravcik, and Milan Hladik. Bounding the support size in extensive form games with imperfect information. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[87] L. Julian Schvartzman and Michael P. Wellman. Stronger CDA strategies through empirical game-theoretic analysis and reinforcement learning. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 249–256, 2009.

[88] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[89] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[90] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[91] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artif. Intell.*, 171(7):365–377, 2007.

[92] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[93] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[94] S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. In *30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

[95] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[96] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE*, 12(4), 2017.

[97] Anderson Tavares, Hector Azpurua, Amanda Santos, and Luiz Chaimowicz. Rock, paper, starcraft: Strategy selection in real-time strategy games. In *The Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-16)*, 2016.

[98] Taylor and Jonker. Evolutionarily stable strategies and game dynamics. *Mathematical Biosciences*, 40:145–156, 1978.

[99] K. Tuyls and R. Westra. Replicator dynamics in discrete and continuous strategy spaces. In *Agents, Simulation and Applications*, pages 218–243. Taylor and Francis, 2008.

[100] Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41–52, 2012.

[101] W. E. Walsh, R. Das, G. Tesauro, and J.O. Kephart. Analyzing complex strategic interactions in multi-agent games. In *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents, 2002.*, 2002.

[102] Michael P. Wellman. Methods for empirical game-theoretic analysis. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[103] S. Whiteson, B. Tanner, M. E. Taylor, and P. Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 120–127, 2011.

[104] James R. Wright and Kevin Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal form games. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 901–907, 2010.

[105] Mason Wright. Using reinforcement learning to validate empirical game-theoretic analysis: A continuous double auction study. *CoRR*, abs/1604.06710, 2016.

[106] Nikolai Yakovenko, Liangliang Cao, Colin Raffel, and James Fan. Poker-CNN: A pattern learning strategy for making draws and bets in poker games using convolutional networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[107] Wako Yoshida, Ray J. Dolan, and Karl J. Friston. Game theory of mind. *PLOS Computational Biology*, 4(12):1–14, 12 2008.

[108] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, 2008.

Appendices

A Meta-Solvers

A.1 Regret matching

Regret matching (RM) is a simple adaptive procedure that leads to correlated equilibria [32], which tabulates cumulative regret $R_i(\pi_{i,k})$ for $\pi_{i,k}$ at epoch k. At each step, for all i simultaneously: $R_i(\pi_{i,k}) \leftarrow R_i(\pi_{i,k}) + u_i(\pi_{i,k}, \sigma_{-i}) - u_i(\sigma)$. A new meta-strategy is obtained by normalizing the positive portions R_i, and setting the negative values to zero:

$$\sigma_i(\pi_{i,k}) = \frac{R^+(\pi_{i,k})}{\sum_k^{[[K]]\cup\{0\}} R^+(\pi_{i,k})} \quad \text{if the denominator is positive,} \quad \text{or} \quad \frac{1}{K+1} \quad \text{otherwise ,}$$

where $x^+ = \max(0, x)$. In our case, we use exploratory strategies that enforce exploration: $\sigma_i' = \gamma\text{UNIF}(K+1) + (1-\gamma)\sigma_i$.

A.2 Hedge

Hedge is similar, except it accumulates only rewards in $x_{i,k}$ and uses a softmax function to derive a new strategy [3]. At each step, $x_{i,k} \leftarrow x_{i,k} + u_i(\pi_{i,k}, \sigma_- i)$, and a new strategy $\sigma_i(\pi_{i,k}) = \exp(\frac{\gamma}{K+1} x_{i,k}) / \sum_k^{[[K]]\cup\{0\}} \exp(\frac{\gamma}{K+1} x_{i,k})$. Again here, we use strategies that mix in $\gamma\text{UNIF}(K+1)$.

A.3 Replicator Dynamics

Replicator dynamics are a system of differential equations that describe how a population of strategies, or replicators, evolve through time. In their most basic form they correspond to the biological *selection* principle, comparing the fitness of a strategy to the average fitness of the entire population. More specifically the symmetric replicator dynamic mechanism is expressed as

$$\frac{dx_k}{dt} = x_k[(\mathbf{Ax})_k - \mathbf{x}^T\mathbf{Ax}].$$

Here, x_k represents the density of strategy $\pi_{i,k}$ in the population ($\sigma_i(\pi_{i,k})$ for a given k), A is the payoff matrix which describes the different payoff values each individual replicator receives when interacting with other replicators in the population. This common formulation represents symmetric games, and typical examples of dynamics are taken from prisoner's dilemma, matching pennies, and stag hunt games. For a more elaborate introduction to replicator dynamics, and their relationship with RL, we refer to [7].

Asymmetric replicator dynamics are applied to n-player games normal form games, *e.g.* in two players using payoff tables A and B, where $A \neq B^T$. Examples are the infamous prisoner's dilemma and the Rock-Scissors-Paper games, in which both agents are interchangeable. In the evolutionary setting this means that the agents are drawn from a single population. In general however, the symmetry assumption no longer holds, as players do not necessarily have access to the same sets of strategies. In this context it means we now have two players that come from different populations:

$$\frac{dx_k}{dt} = x_i[(A\mathbf{y})_k - \mathbf{x}^T A\mathbf{y}], \qquad \frac{dy_k}{dt} = y_i[(\mathbf{x}^T B)_k - \mathbf{x}^T B\mathbf{y}],$$

where x corresponds to the row player and y to the column player. In general, there are n tensors, representing the utility to each player for each outcome.

In this paper we use a new *projected replicator dynamics* that enforces exploration by putting a lower bound on the probability on x_k and y_k.

B Joint Policy Correlation

B.1 Example Comparison Videos

This section points to several example videos of coordination (diagonals of the JPC experiments), and miscoordination (off-diagonals of the JPC experiments).

Laser Tag (small2)

- Diagonal: `https://www.youtube.com/watch?v=8vXpdHuoQH8`
- Off-Diagonal: `https://www.youtube.com/watch?v=jOjwOkCM_i8`

Laser Tag (small3)

- Diagonal: `https://www.youtube.com/watch?v=Z5cpIG3GsLw`
- Off-Diagonal: `https://www.youtube.com/watch?v=zilU0hXvGK4`

In these videos, DQN was used to train the oracle policies, though the effects are similar with ReActor.

B.2 JPC in General n-player Environments

In Section 4.1 we introduced joint policy correlation for symmetric games with $n = 2$ and non-negative rewards. In this section, we present the general description of JPC that can be used for any finite (symmetric or asymmetric) n-player game with arbitrary rewards.

In general, each player has their own tensor of utilities values U_i. If there are d separate independent learning instances, then U_i has dimensionality d^n for each player i, and each entry corresponds to an expected utility to player i receives with a given combination of policies produced for each player in each instance.

For example, for a four-player game and five independent learning instances (labeled $0, 1, \cdots 4$), the value $U_4[0][3][2][2]$ corresponds to the expected return of the fourth player when:

- the first player uses their learned policy π_1 from instance 0,
- the second player uses their learned policy π_2 from instance 3,
- the third player uses their learned policy π_3 from instance 2,
- the fourth player uses their learned policy π_4 from instance 2.

The definitions of average values over diagonals and off-diagonals and average proportional reduction must now be indexed by the player i, and operate only on player i's values in U_i. As a result, $\bar{D}_i$ is an average over d values and $\bar{O}_i$ is an average over $d^n - d$ values, and $\bar{R}_i$ is defined analogously as in Section 4.1 but instead using $\bar{D}_i$ and $\bar{O}_i$. When n is large an estimate of $\bar{O}_i$ can be used instead, by sampling $O(d)$ entries from the exponentially many values.

Note that in asymmetric games, the JPC problem will vary across players. Therefore, it is not clear how to aggregate and report a single value (summary). The simplest solution is to present a vector, $\vec{R}$, containing $\bar{R}_i$ for each player, which exposes how each player is affected separately.

C Algorithm Details and Parameter Values

C.1 Network, Oracle, and Training Parameters

Unless otherwise stated, we use the default parameter settings and architecture reported in the Reactor paper. We set $\lambda = 0.9$, use a LSTM sequence (unroll) length of 32, with a batch size of 4, learning rate $\alpha = 0.0001$, and momentum $\beta_1 = 0$, the ADAM optimizer [49], replay buffer of size 500000, and memorizing behavior probabilities for Retrace off-policy corrections.

The main network architecture is based on the default Reactor network, except without a head for estimated behavior distributions (using purely memorized behavior probabilities), as illustrated in Figure 7.

For gridworld coordination games, we use three convolutional layers of kernel widths (4, 5, 3) and strides (2, 1, 1) each outputting 8 planes. The main fully-connected layer has 32 units and each LSTM layer has 32 units. Every layer, except the LSTM, was followed by a concatenated ReLU layer [89] effectively doubling the number of outputs to 16 for following layers. The rest of the architecture is the same as the default architecture in the Reactor paper.

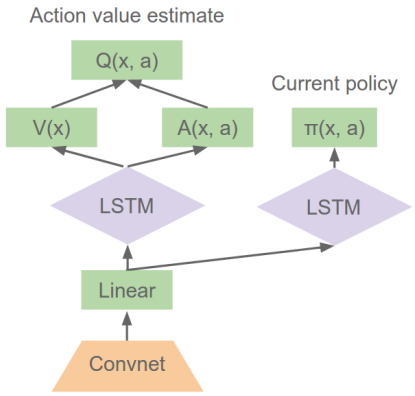


Figure 7: Reactor network architecture. Modified file taken from [31] with permission of authors.

For Leduc poker, we use two fully-connected hidden layers of size 128, with rectified linear units non-linearities at the end of each layer. In each case, these are followed by an LSTM layer of size 32 and fully-connected layer of size 32.

C.2 Neural Fictitious Self-Play (NFSP)

In our tuning experiments for NFSP, we tried and found the following best values for NFSP.

Values of the form $x \to y$ refer to a linear schedule starting at x, ending at (and remaining at) y. Values expresses as 5e-5 refer to $5 \cdot 10^{-5}$.

C.2.1 NFSP parameters for in Two-Player Leduc

Parameter	Value Set	Final Value
RL learning rate	1e-3, 1e-4, 1e-5	1e-3
SL learning rate	1e-3, 1e-4, 1e-5	1e-5
Anticipatory parameter	1e-3, 1e-4, 1e-5	0.1
Reservoir buffer size	100000, 1000000	100000

Best values for NashConv in two-player Leduc.

Parameter	Value Set	Final Value
RL learning rate	1e-3, 1e-4, 1e-5	1e-4
SL learning rate	1e-3, 1e-4, 1e-5	1e-5
Anticipatory parameter	0.1, 0.5	0.5
Reservoir buffer size	100000, 1000000	1000000

Best values for exploitation against random bots in two-player Leduc.

Parameter	Value Set	Final Value
RL learning rate	1e-3, 1e-4, 1e-5	1e-3
SL learning rate	1e-3, 1e-4, 1e-5	1e-4
Anticipatory parameter	0.1, 0.5	0.5
Reservoir buffer size	100000, 1000000	100000

Best values for exploitation against cfr500 bots in two-player Leduc.

C.2.2 NFSP parameters in Three-Player Leduc

Parameter	Value Set	Final Value
RL learning rate	1e-3, 1e-4, 1e-5	1e-4
SL learning rate	1e-3, 1e-4, 1e-5	1e-4
Anticipatory parameter	0.1, 0.5	0.5
Reservoir buffer size	100000, 1000000	1000000

Best values for NashConv in three-player Leduc.

Parameter	Value Set	Final Value
RL learning rate	1e-3, 1e-4, 1e-5	1e-3
SL learning rate	1e-3, 1e-4, 1e-5	1e-5
Anticipatory parameter	0.1, 0.5	0.5
Reservoir buffer size	100000, 1000000	100000

Best values for exploitation versus random bots in three-player Leduc.

Parameter	Value Set	Final Value
RL learning rate	1e-3, 1e-4, 1e-5	1e-5
SL learning rate	1e-3, 1e-4, 1e-5	1e-3
Anticipatory parameter	0.1, 0.5	0.5
Reservoir buffer size	100000, 1000000	100000

Best values for exploitation versus cfr500 bots in three-player Leduc.

C.3 Policy-Space Response Oracles (PSRO)

In our Leduc Poker experiments, we use an alternating implementation that switches periodically between computing the best response (oracle training phase) and meta-strategy learning phase where the empirical payoff tensor is updated and meta-strategy computed. We call this meta-game update frequency in the parameters below. We also use Π^T as described in Section 3, so the current set of policies includes (on the k^{th} epoch) the currently training oracle π'. Finally, we add controlled exploration by linear annealing of the inverse temperature of the softmax policy head, starting at 0 (uniform random) to 1.

Since the setting (action space, observation space, reward space, and network architecture) differ significantly in the setting of Leduc poker, we try different values for the hyper-parameters. Our general methodology was to manually try a few from subsets of sensible ranges on two-player Leduc, then use these values as starting points for three-player. Finally the values in PSRO were used as starting points for DCH parameters.

For each parameter, we also give a rough sensitivity rating on a scale from 1 (not at all sensitive) to 5 (very sensitive) Overall, we found that the algorithms were fairly robust to different parameter values in the range, and we note some main general points: (i) their values differed from those in the visual domains such as Atari and our gridworld games, (ii) the most important parameter was the learning rate.

C.3.1 PSRO Parameters in Two-Player Leduc Poker

Parameter	Value Set	Final Value	SR
Learning rate	5e-5, 1e-4, 2.5e-4, 0.5e-4, 1e-3	5e-4	4
Batch size	4, 8, 16	16	1
Replay buffer size	1e5, 5e5	5e5	1
Sequence (unroll) length	2, 3, 8	2	2
Entropy cost	0, 0.01, 0.1	0.1	2
Exploration decay end	25000, 250000	250000	3
Target update period	1000, 5000, 10000	1000	2
Meta-game update frequency	2500, 5000	2500	1
Episodes per epoch	25000 $\rightarrow$ { 5e4, 1e5, 2.5e5 }	25000 $\rightarrow$ 2.5e5	2
Trace parameter (λ)	0.75, 0.9, 0.95, 1.0	0.95	1
Maximum epochs	20	20	–

SR stands for sensitivity rating. The exploration decay end is in number of steps taken by the ReActor oracle.

C.3.2 PSRO Parameters in Three-Player Leduc Poker

Parameter	Value Set	Final Value	SR
Learning rate	5e-5, 1e-4, 2.5e-4, 0.5e-4, 1e-3	5e-4	4
Batch size	4, 8, 16	16	1
Replay buffer size	1e5, 5e5	5e5	1
Sequence (unroll) length	2, 3, 8	2	2
Entropy cost	0, 0.01, 0.1	0.1	2
Exploration decay end	25000, 250000	250000	3
Target update period	1000, 5000, 10000	1000	2
Meta-game update frequency	2500, 5000	2500	1
Episodes per epoch	25000 $\rightarrow$ { 5e4, 1e5, 2.5e5 }	25000 $\rightarrow$ 2.5e5	2
Trace parameter (λ)	0.75, 0.9, 0.95, 1.0	1.0*	1
Maximum epochs	20	20	–

*The only value that changed moving to three-player Leduc was the trace parameter, λ, and it had such a small effect that in the full runs we left $\lambda = 0.95$ for consistency.

C.4 DCH

C.4.1 DCH Parameters in First-Person Gridworld Coordination Games

In the first-person gridworld games, we use hyper-parameter settings that that are quite similar to the default ReActor values [31], as noted above in Appendix C.1. In this environment, we found that parameter values had little to no effect on the outcomes.

C.4.2 DCH Parameters in Two-Player Leduc Poker

In Leduc poker, there is one new parameter: the policy and meta-strategy save & load frequencies (T_{ms} in Algorithm 2). This is asynchronous analogue to the meta-game update frequency in PSRO. The basic parameter tuning was more difficult for DCH due to large number of resources necessary. Since we want to measure the tension between scalability and accuracy, we tune our hyper parameters only on one value (1000) and include a sweep over $T_{ms} \in \{1000, 2500, 5000\}$ in the full runs. We also try the smaller value since the decoupled meta-solvers are online and require more recent up-to-date estimates of the values.

Parameter	Value Set	Final Value	SR
Learning rate	1e-5, 5e-5, 1e-4, 2.5e-4, 0.5e-4, 1e-3	1e-4	4
Batch size	16	16	–
Replay buffer size	1e5, 5e5	5e5	1
Sequence (unroll) length	2, 3, 8	8	2
Entropy cost	0.1	0.1	–
Exploration decay end	25000, 250000	250000	3
Target update period	1000, 5000, 10000	10000	2
Trace parameter (λ)	0.75, 0.9, 0.95, 1.0	0.95	1
Policy + meta-strategy update frequency	1000, 2500, 5000	2500	1

C.4.3 DCH Parameters in Three-Player Leduc Poker

Parameter	Value Set	Final Value	SR
Learning rate	1e-5, 5e-5, 1e-4, 2.5e-4, 0.5e-4, 1e-3	5e-4	4
Batch size	16	16	–
Replay buffer size	1e5, 5e5	5e5	1
Sequence (unroll) length	2, 3, 8	8	2
Entropy cost	0.1	0.1	–
Exploration decay end	25000, 250000	250000	3
Target update period	1000, 5000, 10000	10000	2
Trace parameter (λ)	0.75, 0.9, 0.95, 1.0	0.95	1
Policy + meta-strategy update frequency	1000, 2500, 5000	2500	1

C.5 Meta-Solvers

For projected replicator dynamics, the average strategy value was tracked using a running average of 50 values. The value of each policy was tracked using a running average of 10 values. The step size was set to $\delta = 0.01$.

D Environments

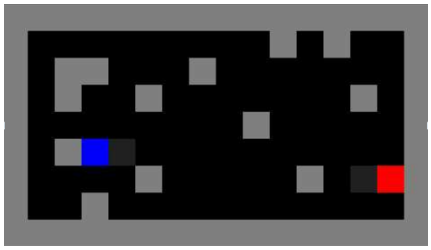


Figure 8: Global view of Laser Tag small3 map (not seen by agents). Agents see a localized view. Input to their networks is raw RGB pixels.

The maps we used are shown in Figure 9.

D.1 Representing Turn-based Games and Handling Illegal Moves

In poker, the number of legal actions that an agent can use is a subset of the total number of unique actions in the entire game. Also, the game is turn-based so only one player acts at a given time.

A modification to the standard environment is made so that policies can be defined over a fixed set of actions and independent of the specific underlying RL algorithm (i.e. DQN, ReAactor, etc.): the environment presents all 3 actions at all times; if an illegal action is taken, then the agent receives a reward equal to the lower bound of the payoff at a terminal node minus 1 ($= -14$ in Leduc), and a random legal move is chosen instead. The resulting game is more complex and the agent must first learn the rules of the game in addition to the strategy. This also makes the game general-sum, so CFR and exploitability were instead run on the original game. Exploitability of PSRO, DCH, and NFSP policies are computed by first transforming the policies to legal ones by masking out illegal moves and renormalizing.

Some RL algorithms process experience using transition tuples of the form *e.g.* (s, a, r, s'), or longer chains such as in ReActor. However, in turn-based games, given a trajectories

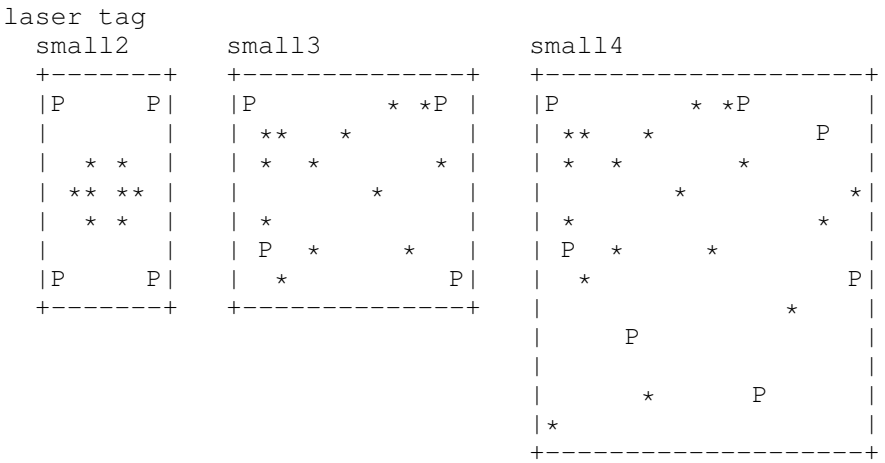
$$(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \cdots),$$

the next state may not belong to the same player, so we construct player-specific tuples of the form $(s_t, a_t, r_t, s_{t+1}, \cdots, s_{t+k}, \cdots)$, where k is the number of steps until it becomes the same player's turn again, so *e.g.* in strictly-alternating games $k = 2$. Special cases are needed for terminal states, where all players see the terminal state as the final transition.

E Results

See the following figures:

- Effect of DCH parameter values on NashConv in two-player Leduc: Figure 10.
- Effect of DCH parameter value of γ on NashConv overall in two-player Leduc: Figure 14.
- Effect of DCH parameter value of γ on NashConv per meta-solver two-player Leduc: Figure 15.
- Effect of DCH parameter values on exploitation in two-player Leduc: Figure 11.
- Exploitation versus cfr500pure in two-player Leduc: Figure 12.
- Exploitation versus random bots in three-player Leduc: Figure 13.



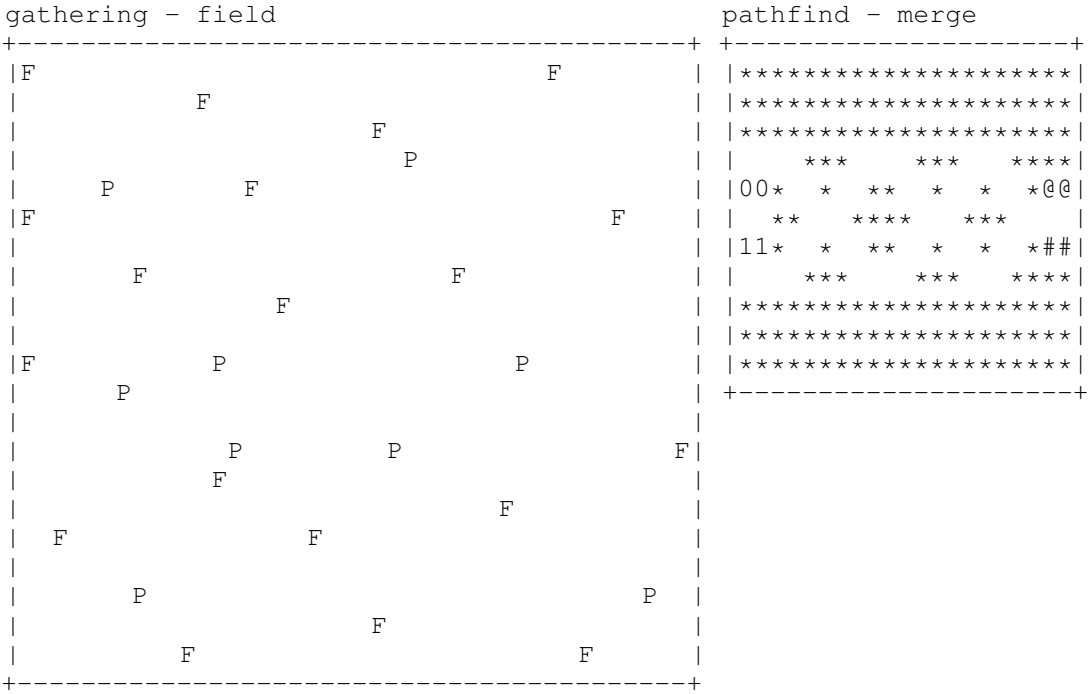


Figure 9: Maps used in gridworld games. Here, `P` is a spawn point for both players, `*` are blocked/wall cells, `F` is food (apples), `0` and `1` are spawn points for player 1 and 2 respectively, `@` and `#` are goal locations for player 1 and 2 respectively.

E.1 Linear Regression Analysis to Interpret the Effect on DCH Parameters

Using the data from the final runs, we also tested the effect of removing individual parameter settings (value of γ, the meta-solver, levels, number of levels, meta-strategy update period, learning rate) on the outcomes of exploitability and explotation in DCH.

We do this by fitting an ordinary least squares model to predict the (exploitability or exploitation) value based on the parameter values, via the Statsmodels Python module [88].

Below, we show the verbatim output of this analysis in Figure 16. What this shows is the effect when removing individual settings of parameter values on the overall prediction that includes all of the data. It does not (necessarily) show the best value for each parameter, since the parameter values could combine in some complex non-linear way.

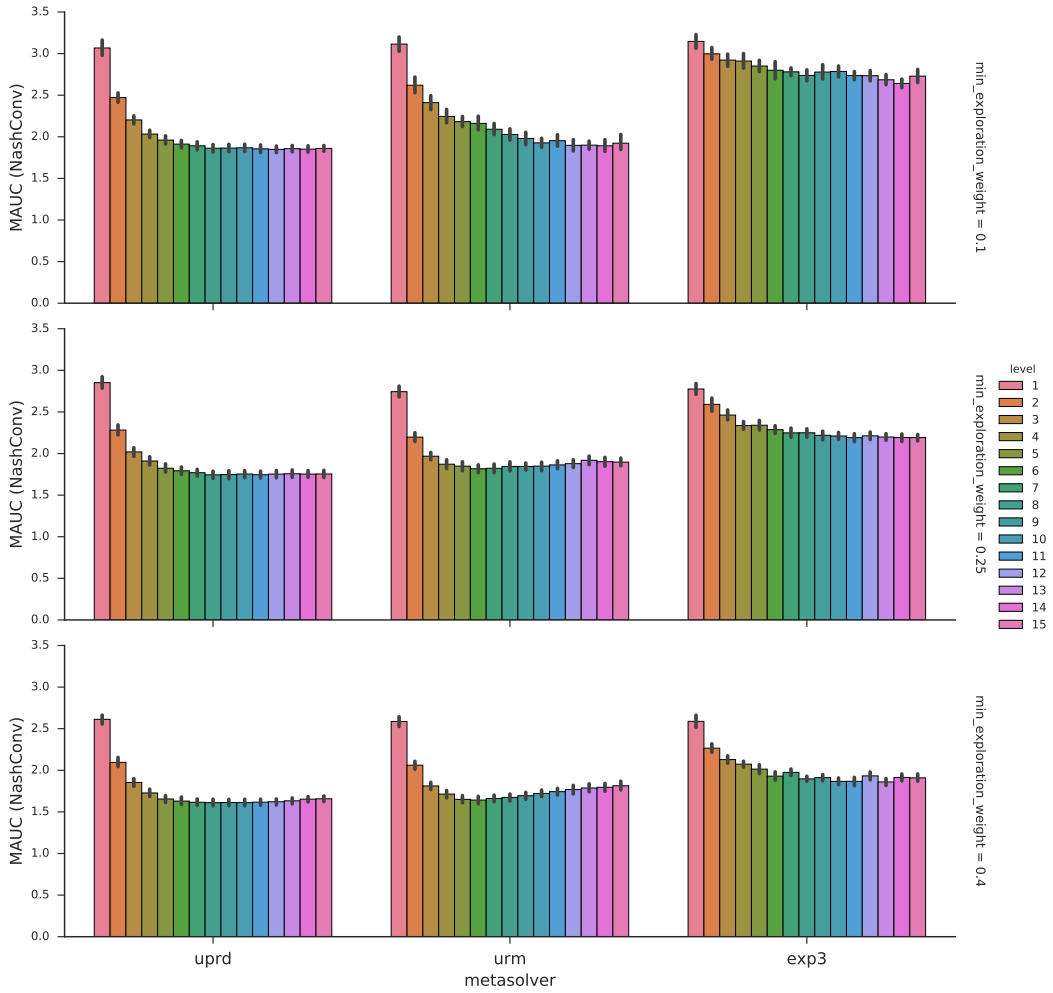


Figure 10: Effect of DCH parameters in two-player Leduc Poker (NashConv).

We observe one particularly interesting point from this analysis: both the meta-solver and the level structure in DCH seem to have a stronger effect on lowering NashConv in three-player Leduc than in two-player Leduc. This could be due to the fact that the two-player game is smaller, and the DCH is struggling to find a precise Nash equilibrium since it is instead maximizing reward against the specific subset of (15) oracles. However, it could also be because opponent modeling and learning to anticipate actions from the other players is more important when learning to play games with more than two players. We hope to investigate this further: particularly the link between training regimes for multiagent reinforcement learning that produce policies capable of generalizing to arbitrary behavior online (during execution), and whether and how this could lead to an implicitly-encoded Theory of Mind.

E.2 CFR Exploitability in Leduc

The convergence graph of vanilla CFR is shown in Figure 17. No abstractions were used.

The NASHCONV values reach at iteration 500 are 0.063591 for two-player, and 0.194337 for three-player.

The value (*i.e.,* expected winnings for first player under any exact Nash equilibrium) of two-player Leduc is -0.085606424078, so the second player has a slight advantage. To compute this number, the exact Nash equilibrium was obtained using sequence-form linear programming [51]; see also [90, Section 5.2.3].

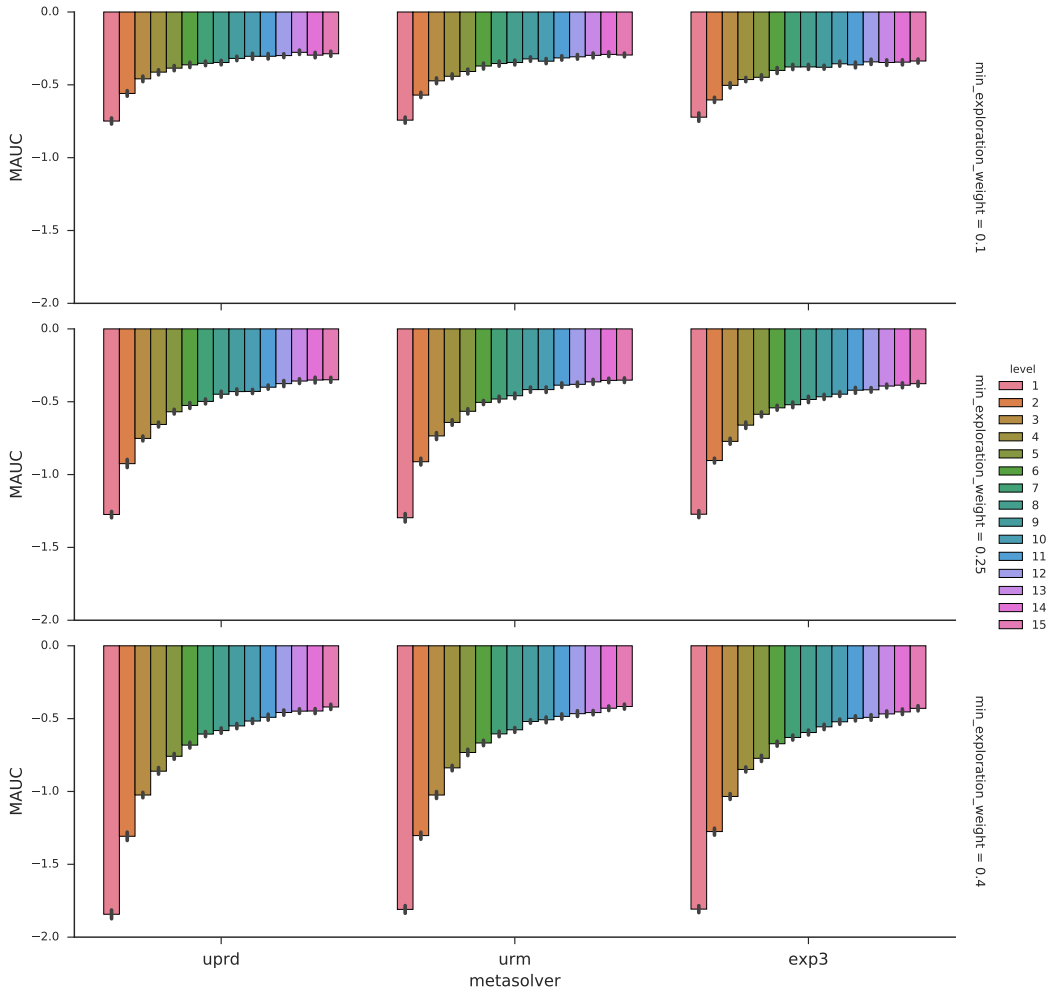


Figure 11: Effect of DCH parameters in two-player Leduc Poker (exploitation).

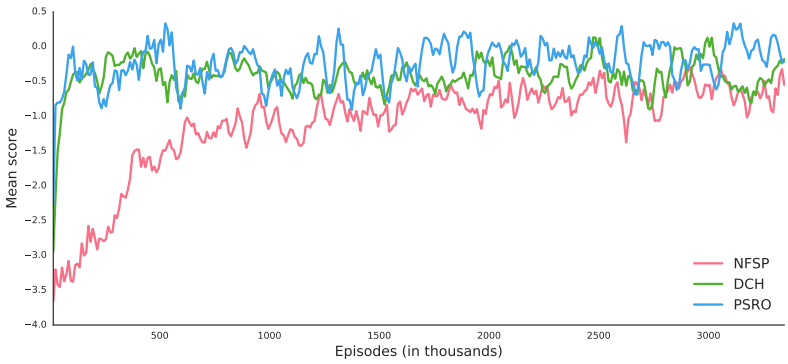


Figure 12: 2-player: Exploitation vs. CFR500pure

E.3 Computing the Explicit Meta-Policy for Exploitability in PSRO/DCH

In PSRO and DCH, the meta-strategy σ_i is a distribution over policies Π_i. The combination of the two encodes a single stochastic policy π_i^σ that can be obtained by doing a pass through the game tree. In terms of computational game theory, this means applying Kuhn's theorem [53] to convert a mixed strategy to a behavior strategy.

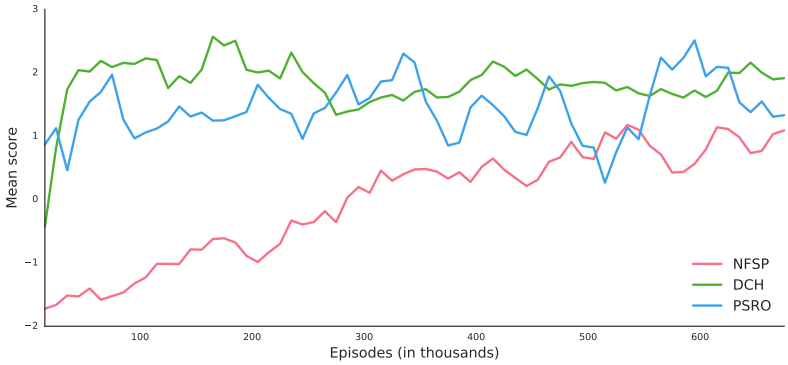


Figure 13: 3-player: exploitation vs. random bots

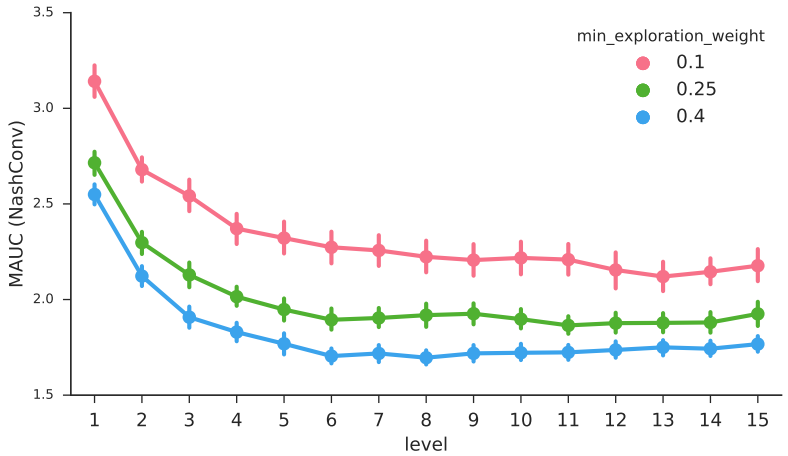


Figure 14: Effect of DCH parameter value of γ on NashConv overall in two-player Leduc

For each information state I, the probability of taking action a is $\pi_i^\sigma(I, a)$. This can be computed by computing weights $W_{I,a}$ which is the sum over global states $s \in I$ of reach probabilities to get to s under each policy $\pi_{i,k} \in \Pi_i$ times the probability $\sigma_i(\pi_{i,k}) \cdot \pi_i(s, a)$. Then the final stochastic policy is obtained by

$$\pi_i^\sigma(s, a) = \frac{W_{I,a}}{\sum_{a'} W_{I,a'}}.$$

(Note that opponents' policies need not be considered in the computation as they would cancel when $W_{I,a}$ is normalized.)

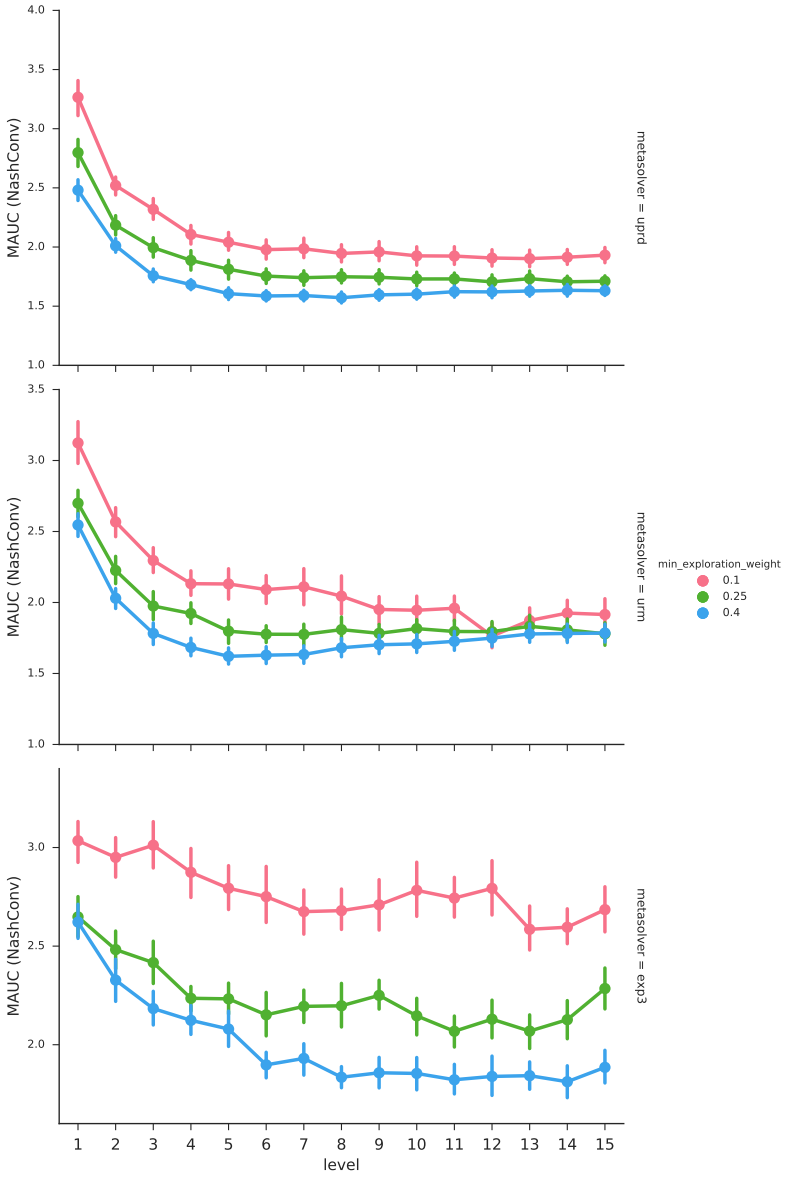


Figure 15: Effect of DCH parameter value of γ on NashConv per meta-solver two-player Leduc

DCH in 2-player Leduc for NashConv
OLS Regression Results

Dep. Variable:	AUC	R-squared:	0.555
Model:	OLS	Adj. R-squared:	0.555
Method:	Least Squares	F-statistic:	897.2
Date:	Mon, 06 Nov 2017	Prob (F-statistic):	0.00
Time:	16:18:25	Log-Likelihood:	-1318.8
No. Observations:	6480	AIC:	2658.
Df Residuals:	6470	BIC:	2725.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.1009	0.015	209.735	0.000	3.072	3.130
C(learning_rate)[T.0.0001]	-0.0551	0.007	-7.472	0.000	-0.070	-0.041
C(min_exploration_weight)[T.0.25]	-0.3261	0.009	-36.109	0.000	-0.344	-0.308
C(min_exploration_weight)[T.0.40]	-0.4791	0.009	-53.049	0.000	-0.497	-0.461
C(metasolver)[T.uprd]	-0.3974	0.009	-44.000	0.000	-0.415	-0.380
C(metasolver)[T.urm]	-0.3359	0.009	-37.195	0.000	-0.354	-0.318
C(load_period)[T.2500]	0.0328	0.009	3.630	0.000	0.015	0.050
C(load_period)[T.5000]	0.0998	0.009	11.047	0.000	0.082	0.117
meta_strategy_update_period	0.0004	0.001	0.384	0.701	-0.002	0.003
np.log(level)	-0.2542	0.005	-52.075	0.000	-0.264	-0.245

Omnibus:	186.685	Durbin-Watson:	0.982
Prob(Omnibus):	0.000	Jarque-Bera (JB):	223.211
Skew:	0.366	Prob(JB):	3.39e-49
Kurtosis:	3.539	Cond. No.	28.3

DCH in 3-player Leduc for NashConv
OLS Regression Results

Dep. Variable:	AUC	R-squared:	0.750
Model:	OLS	Adj. R-squared:	0.750
Method:	Least Squares	F-statistic:	3239.
Date:	Mon, 06 Nov 2017	Prob (F-statistic):	0.00
Time:	16:27:45	Log-Likelihood:	-4302.7
No. Observations:	9720	AIC:	8625.
Df Residuals:	9710	BIC:	8697.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5.7709	0.015	376.465	0.000	5.741	5.801
C(learning_rate)[T.0.0001]	0.2603	0.008	34.046	0.000	0.245	0.275
C(min_exploration_weight)[T.0.25]	-0.4380	0.009	-46.773	0.000	-0.456	-0.420
C(min_exploration_weight)[T.0.40]	-0.6584	0.009	-70.311	0.000	-0.677	-0.640
C(metasolver)[T.uprd]	-0.9475	0.009	-101.178	0.000	-0.966	-0.929
C(metasolver)[T.urm]	-0.9550	0.009	-101.984	0.000	-0.973	-0.937
C(load_period)[T.2500]	-0.0231	0.009	-2.464	0.014	-0.041	-0.005
C(load_period)[T.5000]	0.0757	0.009	8.079	0.000	0.057	0.094
meta_strategy_update_period	-0.0012	0.001	-1.056	0.291	-0.003	0.001
np.log(level)	-0.4797	0.005	-94.801	0.000	-0.490	-0.470

Omnibus:	2064.928	Durbin-Watson:	0.916
Prob(Omnibus):	0.000	Jarque-Bera (JB):	7655.088
Skew:	1.030	Prob(JB):	0.00
Kurtosis:	6.829	Cond. No.	28.3

Figure 16: Output of regression tests to interpret effects of DCH parameters. For these results, we used a MAUC of the NashConv over the most recent 128 values.

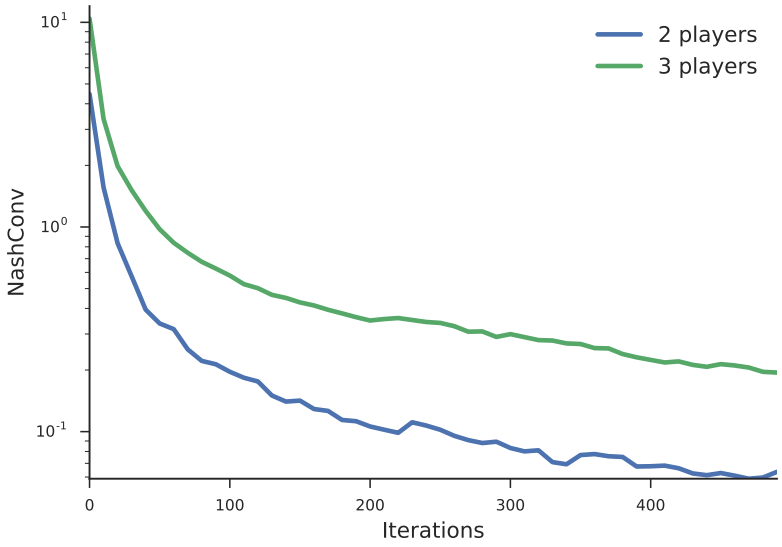


Figure 17: NashConv values of vanilla CFR in two-player and three-player Leduc Poker.