Computation and Learning of Equilibria in Stochastic Games

Noah Golowich

October 15 & 17, 2023

Part I
Upper bounds

1 Independent learning in Stochastic games

We consider a finite-horizon stochastic game $G = (m, \mathcal{S}, \mathcal{A}, \mathbb{P}, r, H, s_1)$. Recall that:

- $m \in \mathbb{N}$ denotes the number of players and $H \in \mathbb{N}$ denotes the horizon.
- $\mathcal{S}$ denotes the state space, which we assume to be finite.
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \mathcal{A}_m$ denotes the joint action set; player i's action set is $\mathcal{A}_i$. We denote joint actions in $\mathcal{A}$ with boldface, i.e., $\mathbf{a} = (a_1, \ldots, a_m) \in \mathcal{A}$.
- $\mathbb{P}(s'|s, \mathbf{a})$ denotes the transition matrix.
- $r_i(s, \mathbf{a})$ denotes the reward function of player i (for $i \in [m]$). We assume $r_i(s, \mathbf{a}) \in [0, 1]$.
- s_1 denotes the initial state. (We assume for simplicity that s_1 is fixed.)

We let $A_i = |\mathcal{A}_i|$ denote the number of actions of player i and $S = |\mathcal{S}|$ denote the number of states.

1.1 Review of notation

Recall that a policy of player i is a sequence of mappings $\pi_{i,h} : (\mathcal{S} \times \mathcal{A})^{h-1} \times \mathcal{S} \to \Delta(\mathcal{A}_i)$, which maps a sequnce of previous states and actions $(s_1, \mathbf{a}_1, \ldots, s_{h-1}, \mathbf{a}_{h-1})$, as well as the current state s_h, to the distribution $\pi_{i,h}(s_1, \mathbf{a}_1, \ldots, s_{h-1}, \mathbf{a}_{h-1}, s_h)$. A *Markov policy of player i* is one which only depends on the step number h and the current state s, i.e., $\pi_{i,h}$ can be written as a function $\pi_{i,h} : \mathcal{S} \to \Delta(\mathcal{A}_i)$.

A *joint policy* is simply a policy specifying actions for all players at each step, i.e., a sequence of mappings $\pi_h : (\mathcal{S} \times \mathcal{A})^{h-1} \times \mathcal{S} \to \Delta(\mathcal{A})$, and a *joint Markov policy* is a sequence $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$, for $1 \leq h \leq H$. A joint policy is a *product policy* if the mappings π_h decompose as $\pi_h : (\mathcal{S} \times \mathcal{A})^{h-1} \times \mathcal{S} \to \Delta(\mathcal{A}_1) \times \cdots \times \Delta(\mathcal{A}_m)$. A joint Markov policy is a product policy if the mappings π_h decompose as $\pi_h : \mathcal{S} \to \Delta(\mathcal{A}_1) \times \cdots \times \Delta(\mathcal{A}_m)$.

Given a policy π, $h \in [H]$, a state $s \in \mathcal{S}$, and a player $i \in [m]$, the player's *value function* at step h is defined by:

$$V_{i,h}^{\pi}(s) := \mathbb{E}^{\pi}\left[\sum_{g=h}^{H} r_i(s_g, \mathbf{a}_g) \;\mid\; s_h = s\right],$$

where the notation $\mathbb{E}^{\pi}[\cdot \mid s_h = s]$ means that we start the game at state s and players play according to policy π.

1.2 Equilibrium notions

Player i's utility is given by their expected value function, i.e., we should think of $u_i(\pi) := V_{i,1}^{\pi}(s_1)$. We will be primarily concerned with *coarse correlated equilibria*, due to their computational tractability:

Definition 1.1 (Coarse correlated equilibrium)**.** A joint policy π is a *ϵ-approximate coarse correlated equilibrium (ϵ-CCE)* if, for each $i \in [m]$,

$$\max_{\pi_i'} V_{i,1}^{\pi_i' \times \pi_{-i}}(s_1) - V_{i,1}^{\pi}(s_1) \leq \epsilon,$$

where $\pi_i' \times \pi_{-i}$ denotes the policy where player i plays according to π_i' and all other players play according to π. If π is also a Markov policy, then it is called an *ϵ-approximate Markov CCE*.

For reference, a Nash equilibrium can be seen as a CCE which is also a product policy:

Definition 1.2 (Nash equilibrium)**.** A joint policy π is an ϵ-approximate Nash equilibrium if it is an ϵ-CCE and is moreover a **product policy**. Similarly, it is an *ϵ-approximate Markov Nash equilibrium* if it is an ϵ-approximate Markov CCE and a product policy.

2 Warm-up: computing Markov CCE

Last lecture we saw that there always exists a Markov Nash equilibrium in finite-horizon stochastic games. The proof was constructive in nature, using backwards induction. However, due to the intractability of computing Nash equilibrium, even in the special case of normal-form games, we do not have an efficient algorithm for computing an (approximate) Markov Nash equilibrium in stochastic games. Algorithm 1 gives an *efficient* algorithm to compute an (approximate) Markov CCE in a stochastic game.

Proposition 2.1. *For any $\epsilon > 0$, Algorithm 1 finds an $\epsilon \cdot H$-approximate Markov CCE of the stochastic game.*

Proof. It is straightforward to show, using backwards induction on h, that for each $h \in [H], i \in [m], s \in \mathcal{S}$, $V_{i,h}^{\pi}(s) = V_{i,h}(s)$, where $V_{i,h}$ are the value functions defined in Algorithm 1.

Next, we show using backwards induction on h that for each $h \in [H], i \in [m], s \in \mathcal{S}$, that

$$\max_{\pi_i'} V_{i,h}^{\pi_i' \times \pi_{-i}}(s) - V_{i,h}^{\pi}(s) \leq \epsilon \cdot (H - h). \tag{1}$$

Algorithm 1 `CCE-Value-Iteration`

```
1: Initialize V_{i,H+1}(s) = 0 for all i ∈ [m], s ∈ S.
2: for h = H, H − 1, ..., 1 do:
3:     for s ∈ S do
4:         For i ∈ [m], define Q_i(s, a) := r_i(s, a) + E_{s'∼P(·|s,a)}[V_{i,h+1}(s')].
5:         Find an ϵ-CCE of the game (Q_1(s, ·), ..., Q_m(s, ·)), denoted q ∈ Δ(A).    ▷ e.g., LP or no-regret
6:         For i ∈ [m], define V_{i,h}(s) := E_{a∼q}[Q_i(s, a)] and π_h(s) := q ∈ Δ(A).
7: return the policy π = (π_1, ..., π_H).
```

Indeed, suppose that (1) holds at step $h+1$. Then for any policy π_i' of player i, and any state $s \in \mathcal{S}$,

$$\begin{aligned}
&V_{i,h}^{\pi_i' \times \pi_{-i}}(s) - V_{i,h}^{\pi}(s) \\
=&\mathbb{E}_{\mathbf{a}\sim\pi_h(s)}\left[\mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,(\pi_i'(s),\mathbf{a}_{-i}))}[r_i(s,(\pi_i'(s),\mathbf{a}_{-i})) + V_{i,h+1}^{\pi_i'\times\pi_{-i}}(s')] - \mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,\mathbf{a})}[r_i(s,\mathbf{a}) + V_{i,h+1}^{\pi}(s')]\right] \\
\leq&\epsilon\cdot(H-h-1) + \mathbb{E}_{\mathbf{a}\sim\pi_h(s)}\left[\mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,(\pi_i'(s),\mathbf{a}_{-i}))}[r_i(s,(\pi_i'(s),\mathbf{a}_{-i})) + V_{i,h+1}^{\pi}(s')] - \mathbb{E}_{s'\sim\mathbb{P}(s,\mathbf{a})}[r_i(s,\mathbf{a}) + V_{i,h+1}^{\pi}(s')]\right] \\
=&\epsilon\cdot(H-h-1) + \mathbb{E}_{\mathbf{a}\sim\pi_h(s)}\left[\mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,(\pi_i'(s),\mathbf{a}_{-i}))}[r_i(s,(\pi_i'(s),\mathbf{a}_{-i})) + V_{i,h+1}(s')] - \mathbb{E}_{s'\sim\mathbb{P}(s,\mathbf{a})}[r_i(s,\mathbf{a}) + V_{i,h+1}(s')]\right] \\
\leq&\epsilon\cdot(H-h-1) + \epsilon = \epsilon\cdot(H-h),
\end{aligned}$$

where the first inequality uses the inductive hypothesis (i.e., (1) at step $h+1$) and the second inequality uses the choice of π_h in Lines 5 and 6 of Algorithm 1.

Finally, note that (1) at step $h=1$ gives the desired result. □

3 The `V-learning` algorithm

Algorithm 1 relies on the assumption that the transitions and rewards of the stochastic game G are known – what if this is not the case? At the same time, we wish to allow the agents to learn in an *independent* manner, not needing to communicate with each other or with a central coordinator. We therefore consider the following *decentralized* setting, where the algorithm does not know the transitions and rewards, and must *learn* them over time:

- The agents are allowed to interact with the stochastic game over a period of T *episodes*.
- At the beginning of each episode $t \in [T]$, each agent $i \in [m]$ chooses a policy $\pi_i = (\pi_{i,1}, \dots, \pi_{i,H})$ (often we will use a superscript as in $\pi_i^{(t)}$ to distinguish policies from different episodes).
- A *trajectory* $(s_h, (a_{1,h}, \dots, a_{m,h}), (r_{1,h}, \dots, r_{m,h}))_{h=1}^H$ is drawn from the joint policy $\pi = (\pi_1, \dots, \pi_m)$, where agents execute π_i independently.
- Each agent i observes $(s_h, a_{i,h}, r_{i,h})_{h=1}^H$ and uses this data to update their policy π_i.

- At the conclusion of the T episodes, each agent i outputs a policy π_i. We consider the 2-player zero-sum setting, and want that the policy (π_1, π_2) is an ϵ-Nash equilibrium (Definition 1.2) (in fact, we will be able to ensure that it is a Markov Nash equilibrium).

  We remark that in the general-sum setting, though we cannot find a Nash equilibrium, it is possible for each player to independently output a policy π_i which depends on some shared random bits, so that a joint policy defined in terms of the π_i and the shared bits is an ϵ-CCE. We do not discuss the details of this procedure.

3.1 The challenge of exploration

We might hope that when the transitions $\mathbb{P}(s'|s, \mathbf{a})$ and rewards $r_i(s, \mathbf{a})$ are unknown, then we can still implement an "approximation" of Algorithm 1, say as follows (sometimes known as an *ϵ-greedy approach*):

1. Draw several trajectories from a random policy and estimating the transitions and rewards from the trajectories.
2. Run Algorithm 1 on the empirical approximation of the transitions and rewards.

It turns out that such a strategy fails. To illustrate, let us consider the $m = 1$ player setting, so that finding an ϵ-approximate equilibrium is equivalent to finding an ϵ-optimal policy.

Example 3.1 (Combination lock)**.** Suppose there are $H+1$ states, denoted $s_1, s_2, \ldots, s_H, s_{\mathsf{sink}}$ and two actions $\mathcal{A} = \{0, 1\}$. Moreover, for some unknown action sequence $a^\star = (a_1, \ldots, a_H) \in \mathcal{A}^H$:

- $\mathbb{P}(s_{h+1}|s_h, a_h^\star) = 1$ and $\mathbb{P}(s_{\mathsf{sink}}|s_h, 1 - a_h^\star) = 1$.
- $r(s_H, a_H^\star) = 1$ and all other rewards are 0.
- The initial state is s_1.

Why does the ϵ-greedy approach fail to learn efficiently on this example? Clearly there is a policy with value 1 (namely, $\pi_h^\star(s_h) = a_h^\star$). However, under a uniformly random policy, the probability that we ever reach state $s_{H/2}$ is $2^{-H/2}$. Thus, in poly(H) trajectories, we will almost never even get to s_h, and certainly have no hope of learning $a_{H/2+1}^\star, \ldots, a_H^\star$.

3.2 The `V-learning` algorithm

For simplicity we assume that all players have A actions, i.e., $|\mathcal{A}_i| = A$ for all i. Moreover, we assume that the stochastic game is 2-player 0-sum, meaning that $m = 2$ and $r_1(s, \mathbf{a}) = 1 - r_2(s, \mathbf{a})$ for all $s \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$.

Example 3.1 indicates that we need to perform some sort of *adaptive exploration*. The `V-learning` algorithm (Algorithm 2) does so by adding larger *exploration bonuses* to states which we have not visited as much. In particular, for a state which has been visited n times in the past, we add an exploration bonus of

$$\beta_n := C\sqrt{H^3 A/n}, \tag{2}$$

for some absolute constant C.[1] `V-learning` performs a sort of "soft" variant of the value iteration procedure of Algorithm 1, where a state s which has been visited n times in the past is updated with a learning rate of

$$\alpha_n := \frac{H+1}{H+n}.$$

Algorithm 2 `V-learning` (for 2-player 0-sum stochastic games)

1: For all i, s, a, h initialize $\widetilde{V}_{i,h}(s), V_{i,h}(s) \leftarrow H, N_h(s) \leftarrow 0, \pi_{i,h}(a|s) \leftarrow 1/A$.

2: For all $(i, s, h) \in [m] \times \mathcal{S} \times [H]$ initialize a bandit no-regret learner $\texttt{BanditNoRegret}_{i,s,h}$ for that tuple.

3: **for** Episode $t = 1, 2, \ldots, T$ **do**

4:     **for** Step $h = 1, 2, \ldots, H$ **do**

5:         **for** Agent $i = 1, 2$ **do**

6:             Take action $a_{i,h} \sim \pi_{i,h}(\cdot|s)$, observe reward $r_{i,h}$ and next state s_{h+1}.

7:             $n \leftarrow N_h(s_h)$, and $N_h(s_h) \leftarrow N_h(s_h) + 1$.

8:             $\widetilde{V}_{i,h}(s_h) \leftarrow (1 - \alpha_n) \cdot \widetilde{V}_{i,h}(s_h) + \alpha_n \cdot (r_{i,h} + V_{i,h+1}(s_{h+1}) + \beta_n)$.

9:             $V_{i,h}(s_h) \leftarrow \min\{H, \widetilde{V}_{i,h}(s_h), V_{i,h}(s_h)\}$.

10:             $\pi_{i,h}(\cdot|s_h) \leftarrow \texttt{BanditNoRegret}_{i,s_h,h}(a_{i,h}, r_{i,h} + V_{i,h+1}(s_{h+1}))$.

11: Define Markov policies $\hat{\pi}_1, \hat{\pi}_2$, where

$$\hat{\pi}_{i,h}(\cdot|s) := \sum_{t=1}^{T} \gamma_{i,t} \cdot \pi_{i,h}^{(t)}(\cdot|s),$$

where $\pi_{i,h}^{(t)}$ is player i's policy from episode t, and $\gamma_{i,t} \in \mathbb{R}_{\geq 0}$ are coefficients (see (3) for the formal definition of $\hat{\pi}_i$).

12: **return** $(\hat{\pi}_1, \hat{\pi}_2)$.

We let $\pi_h^{(t)}, V_{i,h}^{(t)}, \widetilde{V}_{i,h}^{(t)}, N_h^{(t)}, s_h^{(t)}, \mathbf{a}_h^{(t)}$ be the values of the respective objects at the beginning of episode t of `V-learning`.

Background on bandit no-regret learners. `V-learning` makes use of a bandit no-regret learning algorithm at each state, for each agent i (see Line 10), which must choose a distribution over the action set $\mathcal{A}$ (taken to be $\mathcal{A}_i$ in `V-learning`) over multiple rounds. In the abstract setting of bandit no-regret learning, at each round k that the algorithm is called:

- The algorithm chooses a distribution $q^{(k)} \in \Delta(\mathcal{A})$, and samples an action $a^{(k)} \sim q^{(k)}$.
- The adversary chooses a reward vector $u^{(k)} \in [0, 1]^{\mathcal{A}}$.

[1] Technically there should be a log factor, but we ignore log factors.

- The algorithm reveals its action $a^{(k)}$, and receives *noisy* feedback $\widetilde{u}^{(k)}(a^{(k)})$, so that $\mathbb{E}[\widetilde{u}^{(k)}(a^{(k)})|a^{(k)}, u^{(k)}] = u^{(k)}(a^{(k)})$.

The learning algorithm aims to minimize its regret relative to any fixed action:

Lemma 3.1 (Bandit no-regret learning guarantee)**.** *There is a bandit no-regret learning algorithm* `BanditNoRegret` *so that, for any sequence of adversarial utilities $u^{(1)}, \ldots, u^{(K)} \in [0,1]^{\mathcal{A}}$, with high probability, it holds that*

$$\max_{a^\star \in \mathcal{A}} \sum_{k=1}^{K} \left(u^{(k)}(a^\star) - \langle u^{(k)}, q^{(k)} \rangle \right) \leq \tilde{O}(\sqrt{T|\mathcal{A}|}).$$

In the specific case of `V-learning`, the bandit no-regret learning algorithm $\texttt{BanditNoRegret}_{i,s,h}$ is instantiated as follows: fix i, s, h, and suppose that the kth time (s, h) is visited is episode t^k. Then:

- The distribution $q^{(k)}$ is $\pi_{i,h}^{(t^k)}(\cdot|s) \in \Delta(\mathcal{A}_i)$.
- The sampled action $a^{(k)} \sim q^{(k)}$ is $a_{i,h}^{(t_k)}$.
- The reward vector $u^{(k)}$ is defined by $u^{(k)}(a_i) := \mathbb{E}_{\mathbf{a}_{-i} \sim \pi_{-i,h}^{(t^k)}(s)} \left[r_i(s, (a_i, \mathbf{a}_{-i})) + \mathbb{E}_{s' \sim \mathbb{P}(s,(a_i,\mathbf{a}_{-i}))}[V_{i,h+1}^{(t^k)}(s')] \right]$.
- The noisy reward is $\widetilde{u}^{(k)}(a^{(k)}) = r_{i,h}^{(t^k)} + V_{i,h+1}^{(t^k)}(s_{h+1}^{(t^k)})$.

Guarantee for `V-learning`. The `V-learning` algorithm can be shown to output an approximate Markov Nash equilibrium of the stochastic game with high probability:

Theorem 3.2 (`V-learning` guarantee)**.** *In a stochastic game G with horizon H, S states, and A actions per player,* `V-learning` *(Algorithm 2) outputs a policy $\hat{\pi} = (\hat{\pi}_1, \hat{\pi}_2)$ which, with high probability is an ϵ-approximate Markov Nash equilibrium of G for $\epsilon = \tilde{O}(\sqrt{H^5 SA/T})$.*

3.3 Proof of correctness of `V-learning`

Defining the output policy $\hat{\pi}$. First we make the following definitions: for $1 \leq \ell \leq n$,

$$\alpha_0^0 := 0, \qquad \alpha_n^0 := \prod_{j=1}^{n} (1 - \alpha_j), \qquad \alpha_n^\ell := \alpha_\ell \cdot \prod_{j=\ell+1}^{n} (1 - \alpha_j).$$

Think of α_n^ℓ as "the contribution from an update during episode ℓ to the value function during episode n."

Lemma 3.3. *The values α_n^ℓ satisfy the following properties:*

1. $\sum_{\ell=0}^{n} \alpha_n^\ell = 1$.
2. $\max_{i \in [n]} \alpha_n^i \leq 2H/n$.
3. $1/\sqrt{n} \leq \sum_{i=1}^{n} \alpha_n^i / \sqrt{i} \leq 2/\sqrt{n}$.

4. $\sum_{n=i}^{\infty} \alpha_n^i = 1 + 1/H$.

The policies $\hat{\pi}_{i,h}(s)$ are defined as follows: fix $s \in \mathcal{S}, h \in [H]$, and let $t^1, \ldots, t^n$ denote the episodes when (h, s) was visited in the execution of Algorithm 2. Then we set, for $i \in [2]$,

$$\hat{\pi}_{i,h}(\cdot|s) = \sum_{j=1}^{n} \alpha_n^j \cdot \pi_{i,h}^{(t^j)}(\cdot|s). \tag{3}$$

We also will need the following lemma:

Lemma 3.4. *Fix any s, h, t. Suppose that, prior to episode t, (s, h) was previously visited during episodes $t^1 < t^2 < \cdots < t^n < t$. Then*

$$\widetilde{V}_{i,h}^{(t)}(s) = \alpha_n^0 \cdot H + \sum_{j=1}^{n} \alpha_n^j \cdot \left(r_i(s, \mathbf{a}_h^{(t^j)}) + V_{i,h+1}^{(t^j)}(s_{h+1}^{(t^j)}) + \beta_j \right) \tag{4}$$

Proof. The proof is immediate by unpacking the definitions of α_n^i and of $\widetilde{V}_{i,h}^{(t)}(s_h)$ on Line 8 of Algorithm 2. (Note that $\alpha_n^0 = 0$ for $n > 0$ and $\alpha_0^0 = 1$.) □

Optimal value functions. We introduce the following notation: for a function $V_{h+1} : \mathcal{S} \to \mathbb{R}$, we write

$$\mathbb{P}_h V_{h+1}(s, \mathbf{a}) := \mathbb{E}_{s' \sim \mathbb{P}(s,\mathbf{a})}[V_{h+1}(s')].$$

Since (normal-form) 2-player zero-sum games have a well-defined value, 2-player zero-sum stochastic games have well-defined value functions, which are defined iteratively as follows, for $i \in \{1, 2\}$:

$$\begin{aligned} V_{i,h}^{\star}(s) &:= \max_{p \in \Delta(\mathcal{A}_i)} \min_{q \in \Delta(\mathcal{A}_{-i})} \mathbb{E}_{a_i \sim p} \mathbb{E}_{a_{-i} \sim q}[Q_{i,h}^{\star}(s, (a_i, a_{-i}))] \\ Q_{i,h}^{\star}(s, \mathbf{a}) &:= r_{i,h}(s, \mathbf{a}) + (\mathbb{P}_h V_{i,h+1}^{\star})(s, \mathbf{a}). \end{aligned}$$

It is straightforward to show that, for all h, s, i:

$$\max_{\pi_i} \min_{\pi_{-i}} V_{i,h}^{\pi_i, \pi_{-i}}(s) = \min_{\pi_{-i}} \max_{\pi_i} V_{i,h}^{\pi_i, \pi_{-i}}(s) = V_{i,h}^{\star}(s). \tag{5}$$

Moreover, we note that the following equality (called the *Bellman equation*) is immediate from the definition of $V_{i,h}^{\pi}$:

$$V_{i,h}^{\pi}(s) = \mathbb{E}_{\mathbf{a} \sim \pi_h(s)} \left[r_i(\mathbf{a}) + \mathbb{P}_h V_{i,h+1}^{\pi}(s, \mathbf{a}) \right]. \tag{6}$$

Optimism. The following lemma shows that the exploration bonuses induce value functions which are overestimates of the optimal value function:

Lemma 3.5. *With high probability, for all $s \in \mathcal{S}, h \in [H], t \in [T], i \in [2]$,*

$$\widetilde{V}_{i,h}^{(t)}(s) \geq V_{i,h}^{(t)}(s) \geq \max_{\pi_i'} V_{i,h}^{\pi_i', \hat{\pi}_{-i}}(s) \geq V_{i,h}^{\star}(s).$$

Proof. The first inequality is immediate from the definition of $V_{i,h}^{(t)}$ in Line 9, and the final inequality is immediate from (5). So it remains to only show the middle inequality. So fix s, h, t, and suppose that, prior to episode t, (s, h) was visited in episodes $t^1 < \cdots < t^n < t$. Fix any policy of player i, π_i'. We may also suppose that $n > 0$. Then

$$\begin{aligned}
V_{i,h}^{\pi_i', \hat{\pi}_{-i}}(s) =& \sum_{j=1}^{n} \alpha_n^j \cdot \mathbb{E}_{\mathbf{a} \sim \pi_{i,h}'(s) \times \pi_{-i,h}^{(t^j)}(s)} \left[\left(r_i + (\mathbb{P}_h V_{i,h+1}^{\pi_i', \hat{\pi}_{-i}}) \right) (s, \mathbf{a}) \right] \\
\leq & \sum_{j=1}^{n} \alpha_n^j \cdot \mathbb{E}_{\mathbf{a} \sim \pi_{i,h}'(s) \times \pi_{-i,h}^{(t^j)}(s)} \left[\left(r_i + (\mathbb{P}_h V_{i,h+1}^{(t^j)}) \right) (s, \mathbf{a}) \right] \\
\leq & \sum_{j=1}^{n} \alpha_n^j \cdot \mathbb{E}_{\mathbf{a} \sim \pi_{i,h}^{(t^j)}(s) \times \pi_{-i,h}^{(t^j)}(s)} \left[\left(r_i + (\mathbb{P}_h V_{i,h+1}^{(t^j)}) \right) (s, \mathbf{a}) \right] + \widetilde{O}(\sqrt{H^3 A/n}) \\
\leq & \sum_{j=1}^{n} \alpha_n^j \cdot \left(r_i(s, \mathbf{a}_h^{(t^j)}) + V_{i,h+1}^{(t^j)}(s_{h+1}^{(t^j)}) \right) + \widetilde{O}(\sqrt{H^3 A/n}) \\
=& \widetilde{V}_{i,h}^{(t)}(s),
\end{aligned} \tag{7}$$

where the equality uses (6) together with the definition of $\hat{\pi}_{-i,h}$ in (3), the first inequality uses the inductive hypothesis, the second inequality uses the guarantee of $\texttt{BanditNoRegret}_{i,s,h}$ (namely, Lemma 3.1),[2] the third inequality uses the Azuma-Hoeffding bound and holds with high probability,[3] and the final equality uses Lemma 3.4 and the definition of β_n in (2).

It is straightforward from Line 9 that

$$V_{i,h}^{(t)}(s) = \min \left\{ H, \min_{t' \leq t} \widetilde{V}_{i,h}^{(t')}(s) \right\},$$

Since (7) holds for any t, we conclude that $V_{i,h}^{\pi_i', \hat{\pi}_{-i}}(s) \leq V_{i,h}^{(t)}(s)$, which concludes the proof of the lemma. □

Now we are ready to prove Theorem 3.2.

Proof of Theorem 3.2. It suffices to show that $\max_{\pi_1'} V_{1,1}^{\pi_1', \hat{\pi}_2}(s_1) - \min_{\pi_2'} V_{1,1}^{\hat{\pi}_1, \pi_2'}(s_1) \leq \epsilon$. We may compute

$$\begin{aligned}
\max_{\pi_1'} V_{1,1}^{\pi_1', \hat{\pi}_2}(s_1) - \min_{\pi_2'} V_{1,1}^{\hat{\pi}_1, \pi_2'}(s_1) =& \max_{\pi_1'} V_{1,1}^{\pi_1', \hat{\pi}_2}(s_1) + \max_{\pi_2'} V_{2,1}^{\hat{\pi}_1, \pi_2'}(s_1) - H \\
\leq & V_{1,1}^{(T)}(s_1) + V_{2,1}^{(T)}(s_1) - H \\
\leq & \frac{1}{T} \sum_{t=1}^{T} \left(V_{1,1}^{(t)}(s_1) + V_{2,1}^{(t)}(s_1) \right) - H \\
\leq & \frac{1}{T} \sum_{t=1}^{T} \left(\widetilde{V}_{1,1}^{(t)}(s_1) + \widetilde{V}_{2,1}^{(t)}(s_1) \right) - H,
\end{aligned}$$

[2] Technically, Lemma 3.1 is not quite sufficient: we need to analyze a *weighted* version of regret, where iteration j is weighted by α_n^j; see [JLWY21, Corollary 19] for the correct version.

[3] To apply Azuma-Hoeffding we use Item 2 of Lemma 3.3.

where the first equality uses that $r_1(s, \mathbf{a}) + r_2(s, \mathbf{a}) = H$ for all $s, \mathbf{a}$, the first inequality uses Lemma 3.5, the second inequality uses that $V_{i,1}^{(t)}$ is non-increasing with respect to t (Line 9), and the final inequality also uses Line 9.

Define $\delta_h^{(t)} := \widetilde{V}_{1,h}^{(t)}(s_h^{(t)}) + \widetilde{V}_{2,h}^{(t)}(s_h^{(t)}) - (H - h + 1)$. Note that

$$\delta_h^{(t)} \geq V_1^\star(s_h^{(t)}) + V_2^\star(s_h^{(t)}) - (H - h + 1) = 0,$$

where the inequality uses Lemma 3.5. We now bound $\delta_h^{(t)}$ using forward induction on h, as follows: fix any $h \in [H], t \in [T]$. Let $s = s_h^{(t)}, n_h^{(t)} = N_h^{(t)}(s)$ denote the number of times (h, s) was visited prior to episode t, and let those episodes be denoted $t^1 < t^2 < \cdots < t^n < t$. Then

$$\begin{aligned}
\delta_h^{(t)} =& 2\alpha^0_{n_h^{(t)}} \cdot H + \sum_{j=1}^{n_h^{(t)}} \alpha^j_{n_h^{(t)}} \cdot \left(V_{1,h+1}^{(t^j)}(s_{h+1}^{(t^j)}) + V_{2,h+1}^{(t^j)}(s_{h+1}^{(t^j)}) + (H - h) + 2\beta_j \right) \\
\leq& 2\alpha^0_{n_h^{(t)}} \cdot H + \sum_{j=1}^{n_h^{(t)}} \alpha^j_{n_h^{(t)}} \cdot \left(\widetilde{V}_{1,h+1}^{(t^j)}(s_{h+1}^{(t^j)}) + \widetilde{V}_{2,h+1}^{(t^j)}(s_{h+1}^{(t^j)}) + (H - h) + 2\beta_j \right) \\
\leq& 2\alpha^0_{n_h^{(t)}} \cdot H + \sum_{j=1}^{n_h^{(t)}} \alpha^j_{n_h^{(t)}} \cdot \delta_{h+1}^{(t^j)} + \widetilde{O}(\sqrt{H^3 A / n_h^{(t)}}),
\end{aligned} \tag{8}$$

where the first equality uses Lemma 3.4 for $i = 1, 2$ as well as the fact that $r_1(s, \mathbf{a}) + r_2(s, \mathbf{a}) = 1$ for all $s, \mathbf{a}$, the first inequality uses the fact that $V_{i,h}^{(t)}(s) \leq \widetilde{V}_{i,h}^{(t)}(s)$ for all i, h, s (Line 9), and the final inequality uses the definition of β_j in (2) as well as Item 3 of Lemma 3.3.

We now want to sum (8) over all t. We consider the first two terms separately:

$$\sum_{t=1}^T 2\alpha^0_{n_h^{(t)}} \cdot H = \sum_{t=1}^T 2H \cdot \mathbb{1}\{n_h^{(t)} = 0\} \leq 2HS$$

$$\sum_{t=1}^T \sum_{j=1}^{n_h^{(t)}} \alpha^j_{n_h^{(t)}} \cdot \delta_{h+1}^{(t^j)} \leq \sum_{t'=1}^T \delta_{h+1}^{(t')} \sum_{\ell = n_h^{(t')}+1}^{\infty} \alpha_\ell^{n_h^{(t')}} \leq \left(1 + \frac{1}{H}\right) \cdot \sum_{t'=1}^T \delta_{h+1}^{(t')}.$$

In the second line, the first inequality follows by, for each t', grouping together all terms of the summation of the form $\alpha^j_{n_h^{(t)}}$ for which $j = n_h^{(t')}$. We have terms for visiting the corresponding state $s_h^{(t')}$ for the $n_h^{(t')} + 1, n_h^{(t')} + 2, \ldots$th times. The second inequality uses Item 4 of Lemma 3.3. Note that both inequalities use non-negativity of $\delta_h^{(t)}$.

Using the above display in (8), we conclude that

$$\sum_{t=1}^T \delta_1^{(t)} \leq 2SAH + \left(1 + \frac{1}{H}\right) \cdot \sum_{t=1}^T \delta_{h+1}^{(t)} + \sum_{t=1}^T \widetilde{O}\left(\sqrt{H^3 A / n_h^{(t)}}\right).$$

Iterating on h, we conclude that

$$\begin{aligned} T \cdot \left(\max_{\pi_1'} V_{1,1}^{\pi_1', \hat{\pi}_2}(s_1) - \min_{\pi_2'} V_{1,1}^{\hat{\pi}_1, \pi_2'}(s_1) \right) \leq \sum_{t=1}^{T} \delta_h^{(t)} &\leq O(SAH^2) + \widetilde{O}\left(\sum_{h=1}^{H} \sum_{t=1}^{T} \sqrt{H^3 A / n_h^{(t)}} \right) \\ &\leq O(SAH^2) + \widetilde{O}\left(\sum_{s \in \mathcal{S}, h \in [H]} \sum_{n=1}^{N_h^{(T)}(s)} \sqrt{H^3 A / n} \right) \\ &\leq O(SAH^2) + \sqrt{T \cdot H^5 SA}. \end{aligned}$$

$\square$

Part II

Lower bounds

4 Hardness results for no-regret learning

Recall from last lecture we saw the `V-learning` algorithm. While we focused on the 2-player 0-sum setting, in which `V-learning` finds a Markov Nash equilibrium of a given stochastic game, there is a more general version of the algorithm that works in multi-player, general-sum stochastic games:

1. Over the course of T episodes, players independently choose policies $\pi_i^{(t)} = (\pi_{i,1}^{(t)}, \ldots, \pi_{i,h}^{(t)})$ (using essentially the same procedure as in `V-learning`).
2. At the end of the T episodes, the players output a *joint* policy $\hat{\pi}$, which is an approximate CCE. It takes some coordination to act according to $\hat{\pi}$ (i.e., the players need access to a string of shared random bits).

A more natural guarantee (and in line with classical work on no-regret learning we see elsewhere in the course) would be the following:

Question 4.1. Do the policies $\hat{\pi}_i^{(t)}$ ($i \in [m], t \in [T]$) produced by `V-learning` satisfy the following *no-regret* guarantee: for each $i \in [m]$,

$$\max_{\pi_i'} \sum_{t=1}^{T} V_{i,1}^{\pi_i' \times \pi_{-i}^{(t)}}(s_1) - V_{i,1}^{\pi^{(t)}}(s_1) \leq o(T).$$

It turns out that the answer that Question 4.1 is no, in a fairly strong sense:

Theorem 4.1 ([FGK23]). *If* $\mathsf{PPAD} \not\subseteq \mathsf{RP}$,[4] *then there is no polynomial-time algorithm which takes as input the description of a stochastic game and outputs a sequence of joint product policies that guarantees each agent sublinear regret.*

5 Hardness results for computing stationary Markov equilibria

Note that the equilibria we computed and learned in the previous lecture were *nonstationary* equilibria, meaning that the actions players take depends on the time step h. This is necessary in a finite-horizon setting: in general, there may not be a Nash equilibrium in which players' policies are stationary. However, as we saw two lectures ago, in the *discounted infinite-horizon setting*, there always exists a Nash equilibrium in stationary strategies. Below we review the relevant notions:

We consider a discounted infinite-horizon stochastic game $G = (m, \mathcal{S}, \mathcal{A}, \mathbb{P}, r, \gamma)$, which is defined identically to the finite-horizon setting, except that the horizon H is replaced by a *discount factor* $\gamma \in (0, 1)$. Recall that $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$. Also note that larger γ (i.e., closer to 1) represents harder problems.

[4] We use RP to denote the class of total search problems in TFNP for which there is a polynomial-time algorithm which outputs a solution with probability at least $2/3$.

5.1 Policies and equilibria

A *(Markov) joint stationary policy*[5] is a mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The policy π is a product policy if it decomposes as $\pi : \mathcal{S} \to \Delta(\mathcal{A}_1) \times \cdots \times \Delta(\mathcal{A}_m)$. A *(Markov) stationary policy of player $i \in [m]$* is a mapping $\pi_i : \mathcal{S} \to \Delta(\mathcal{A}_i)$.

The value functions are defined as follows in the discounted infinite-horizon setting: for a joint policy π, a state $s \in \mathcal{S}$, and a player $i \in [m]$, we define

$$V_i^\pi(s) := \mathbb{E}^\pi \left[\sum_{h=1}^{\infty} \gamma^{h-1} \cdot r_i(s_h, \mathbf{a}_h) \;\mid\; s_1 = s \right].$$

Note that $V_i^\pi(s) \in [0, 1/(1-\gamma)]$ (since $r_i(s, \mathbf{a}) \in [0,1]$). In this section we will always consider the case that γ is a constant, meaning that $V_i^\pi(s) \in [0, O(1)]$.

Definition 5.1 (Stationary CCE)**.** A stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is an *ϵ-approximate stationary CCE* if for all states $s \in \mathcal{S}$ and players $i \in [m]$,

$$\max_{\pi_i'} V_i^{\pi_i' \times \pi_{-i}}(s) - V_i^\pi(s) \le \epsilon. \tag{9}$$

Definition 5.2 (Stationary Nash equilibrium)**.** A joint stationary policy π is an *ϵ-approximate stationary Nash equilibrium* if it is an ϵ-approximate stationary CCE and is moreover a **product policy**.

Note that Definitions 5.1 and 5.2, as stated above, are a bit stronger than their counterparts in the nonstationary setting, in the sense that the nondeviation conditions (9) must hold for *all* states (not just the initial state). This is not such a big deal: there is a (lossy) equivalence between these two notions, and details may be found in [DGZ23].[6] Moreover, the proof from earlier on in the course that there always exists a stationary Nash equilibrium (and thus a stationary CCE) extends immediately to this stronger notion of equilibrium.

5.2 Hardness of stationary CCE

Surprisingly, in contrast to the nonstationary setting, it is computationally hard to compute an ϵ-approximate stationary CCE in a stochastic game (even if all the transitions and rewards are known).

Theorem 5.1. *There is a constant $\epsilon_0 > 0$ so that the problem of computing ϵ_0-approximate stationary CCE in 2-player stochastic games is* PPAD*-hard.*

Theorem 5.1 should be somewhat surprising, in light of the fact that typically it is tractable to compute CCE. To elucidate the source of hardness, we define *turn-based games*, which is a subset of stochastic games that contains the hard instances used to prove Theorem 5.1.

Definition 5.3 (Turn-based game)**.** A discounted infinite-horizon stochastic game is a *turn-based (stochastic) game* if, for each state s, there is some player $i \in [m]$, called the *controller* of the state, so that all players' rewards at s and the transition to the next state from s only depend on the action of player i at s. We write $i = \mathsf{cr}(s)$.

[5] All policies considered henceforth will be Markov, so we drop the "Markov" modifier.

[6] Note also that the algorithm `CCE-Value-Iteration` for computing nonstationary Markov CCE (Algorithm 1) in fact computes the stronger notion in which the nondeviation condition holds at each state.

The crucial observation is the following:

Lemma 5.2. *In turn-based stochastic games, given an ϵ-approximate stationary CCE, an ϵ-approximate stationary Nash equilibria may be constructed in polynomial time.*

Proof. Given a stationary CCE $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, define $\widetilde{\pi} : \mathcal{S} \to \Delta(\mathcal{A}_1) \times \cdots \times \Delta(\mathcal{A}_m)$ by $\widetilde{\pi}(s) := \pi_1(s) \times \cdots \times \pi_m(s)$, where $\pi_i(s)$ denotes the marginal of $\pi(s)$ on player i's actions. Clearly $\widetilde{\pi}$ is a product policy. Since at each state s, the marginals of $\widetilde{\pi}(s)$ and $\pi(s)$ on player $\mathsf{cr}(s)$'s actions agree, it is immediate that $\widetilde{\pi}$ induces the same distribution over trajectories as π and thus is an ϵ-approximate stationary Nash equilibrium. ☐

By Lemma 5.2, it suffices to show that finding ϵ_0-stationary Nash equilibria is PPAD-hard. To do so, we use a similar gadget construction and reduction from an arithmetic circuit problem as was used to show hardness of computing Nash equilibria in graphical games. In particular, we consider the following `Generalized Circuit` problem, which is a variant of the `ArithmCircuitSAT` [DGP06] we saw when showing PPAD-hardness of computing Nash equilibria in normal form games.

Definition 5.4 (`Generalized Circuit` problem)**.** The input to a `Generalized Circuit` problem instance is a circuit with a set V of nodes, together with a set $\mathcal{G}$ of gates connecting the nodes. Each gate is one of the following types: $G_+, G_\times, G_\leftarrow, G_>$. An assignment $\pi : V \to [0,1]$ of real values to the nodes is said to be an *ϵ-approximate solution* to the instance if:

- For each G_+ gate with inputs $u_1, u_2 \in V$ and output $v \in V$, $\pi(v) = \min\{1, \pi(u_1) + \pi(u_2)\} \pm \epsilon$.
- Each $G_\times$ gate comes equipped with a parameter $\alpha \in [-1, 1]$. For each such gate with input $u \in V$ and output $v \in V$, $\pi(v) = \max\{0, \alpha \cdot \pi(u)\} \pm \epsilon$.
- Each $G_\leftarrow$ gate comes equipped with a parameter $\zeta \in [0, 1]$. For each such gate with output $v \in V$, $\pi(v) = \zeta \pm \epsilon$.
- For each $G_>$ gate with inputs $u_1, u_2 \in V$ and output $v \in V$, $\pi(v) = \begin{cases} 1 \pm \epsilon & : \pi(u_1) \geq \pi(u_2) + \epsilon \\ 0 \pm \epsilon & : \pi(u_1) \leq \pi(u_2) - \epsilon \end{cases}$.

The computational problem ϵ-`Generalized Circuit` is as follows: given a circuit $(V, \mathcal{G})$ as described above, to find an ϵ-approximate solution π.

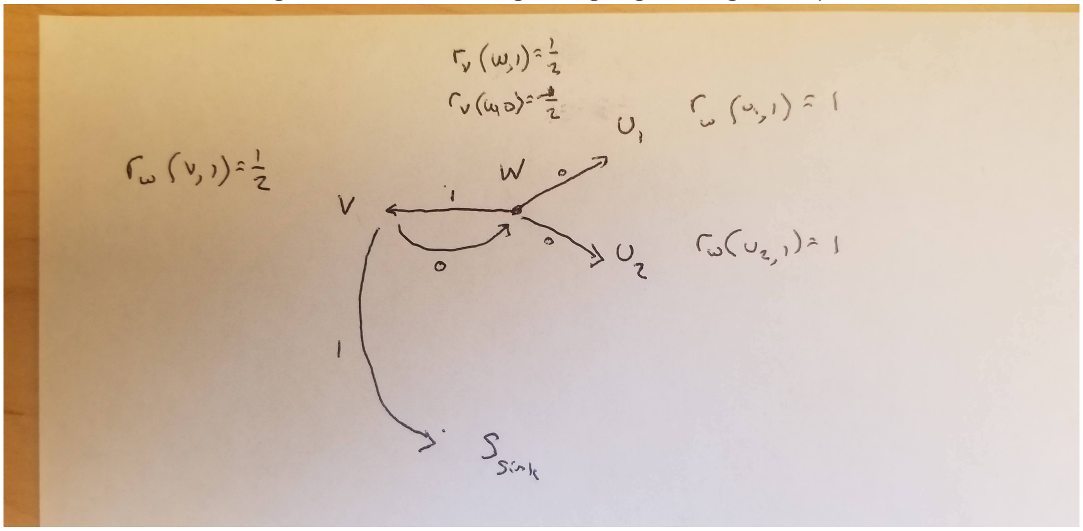
Theorem 5.3 ([Rub18])**.** *There is a constant $\epsilon > 0$ so that ϵ-`Generalized Circuit` is PPAD-hard.*

We are now ready to give the proof (sketch) of Theorem 5.1.

Proof sketch of Theorem 5.1. Choose ϵ according to Theorem 5.3, and let $(V, \mathcal{G})$ denote an instance of ϵ-`Generalized Circuit`. We will construct a turn-based stochastic game $\mathbb{G}$ so that, for some constant $\epsilon_0 < \epsilon$, any ϵ_0-approximate stationary Nash equilibrium of $\mathbb{G}$ allows us to compute an ϵ-approximate solution to $(V, \mathcal{G})$. The game $\mathbb{G}$ is constructed as follows:

- The state space is $\mathcal{S} = V \cup W \cup \{s_{\mathsf{sink}}\}$, where s_{sink} is a "sink state" which transitions to itself and which yields a reward of 0 to all players, and W is a set of "helper nodes" (discussed further below).

Figure 1: Stochastic game gadget for gate G_+.



- The set of players is $V \cup W$: each player controls a single state.[7]
- The action set of each player is $\mathcal{A}_i = \{0, 1\}$.
- The rewards $r_i(s, a_{\mathsf{cr}(s)})$ and transitions $\mathbb{P}(s'|s, a_{\mathsf{cr}(s)})$ are specified below, as a function of $(V, \mathcal{G})$. Note that when specifying rewards and transitions it suffices to specify only the action of the controller $\mathsf{cr}(s)$ of s, since the game is turn-based.
- The discount factor is $\gamma = \epsilon^2$.

The idea is now to construct the rewards and transitions of $\mathbb{G}$ to "simulate" each gate in $\mathcal{G}$. To illustrate, we consider a gate of the form G_+, with input nodes $u_1, u_2 \in V$ and output node $v \in V$. To simulate this gate, we need one helper node $w \in W$. The transitions out of w and v are defined as follows:

- $\mathbb{P}(u_1|w, 0) = \mathbb{P}(u_2|w, 0) = \frac{1}{2}$; $\mathbb{P}(v|w, 1) = 1$.
- $\mathbb{P}(w|v, 0) = 1$ and $\mathbb{P}(s_{\mathsf{sink}}|v, 1) = 1$.

The rewards to the players $\mathsf{cr}(v), \mathsf{cr}(w)$ controlling states v, w are as follows:

- $r_{\mathsf{cr}(w)}(u_1, 1) = r_{\mathsf{cr}(w)}(u_2, 1) = 1$, and $r_{\mathsf{cr}(w)}(v, 1) = 1/2$.
- $r_{\mathsf{cr}(v)}(w, 1) = 1/2$ and $r_{\mathsf{cr}(v)}(w, 0) = -1/2$.
- All other rewards to $\mathsf{cr}(w), \mathsf{cr}(v)$ are 0.

See Figure 1. Since $\mathbb{G}$ is a turn-based game, any product stationary policy π corresponds to a mapping $\pi : \mathcal{S} \to [0, 1]$, where $\pi(s) = \pi_{\mathsf{cr}(s)}(1|s)$ is the probability that the controller of s takes action 1 at s. The following lemma shows that the transitions and rewards defined above simulate the gate G_+:

[7] Note that Theorem 5.1 states PPAD-hardness for 2-player games, which is a bit stronger than the result we prove. This construction may be modified to only have 2 players, though the proof is more complicated; see [DGZ23].

Lemma 5.4. *There is a constant $\epsilon_0 \ll \epsilon$, so that the following holds.*[8] *For any ϵ_0-approximate stationary Nash equilibrium of $\mathbb{G}$, it holds that*

$$\pi(v) = \min\{1, \pi(u_1) + \pi(u_2)\} \pm \epsilon.$$

The proof of Lemma 5.4 is provided below. Similar constructions and lemmas may be provided for the other gate types $G_\times, G_\leftarrow, G_>$. Putting those results together, it follows that, given an ϵ_0-approximate stationary Nash equilibrium π of $\mathbb{G}$, we may consider the corresponding mapping $\pi : \mathcal{S} \to [0, 1]$, and its restriction to V yields an ϵ-approximate solution to $(V, \mathcal{G})$, as desired. $\square$

To prove Lemma 5.4, we need the following notation: for $s \in \mathcal{S}$, $a \in \mathcal{A} = \{0, 1\}$, and $i \in [m]$,

$$Q_i^\pi(s, a) := \mathbb{E}^\pi \left[\sum_{h=1}^{\infty} \gamma^{h-1} \cdot r_i(s_h, \mathbf{a}_h) \ \middle| \ s_1 = s,\ a_{1,\mathsf{cr}(s)} = a \right].$$

We need the following lemma:

Lemma 5.5. *Given an ϵ_0-approximate stationary Nash equilibrium π of $\mathbb{G}$, it may be converted in polynomial time to a stationary policy $\bar{\pi}$ which satisfies: for all $i \in [m], s \in \mathcal{S}$,*

$$\max_{a \in \mathcal{A}} Q_i^{\bar{\pi}}(s, a) - \min_{a' \in \mathcal{A} : \bar{\pi}_{\mathsf{cr}(s)}(a'|s) > 0} Q_i^{\bar{\pi}}(s, a') \leq \epsilon', \tag{10}$$

for some $\epsilon' = O(\sqrt{\epsilon_0})$.

The policy $\bar{\pi}$ is known as an *ϵ-well supported stationary Nash equilibrium of* $\mathbb{G}$: roughly saying, (10) is saying that for any action on which $\bar{\pi}$ puts nonzero probability, it must be almost as good (i.e., within ϵ') of the optimal action at that state. See [DGZ23, Lemmas 5.5 & 5.6] for a proof of Lemma 5.5.

Proof sketch of Lemma 5.4. First let $\bar{\pi}$ be the output of running the procedure of Lemma 5.5 on π. Suppose that $\bar{\pi}(v) > \bar{\pi}(u_1) + \bar{\pi}(u_2) + \epsilon$. Then using the definition of $Q_i^{\bar{\pi}}$, we have

$$Q_{\mathsf{cr}(w)}^{\bar{\pi}}(w, 1) - Q_{\mathsf{cr}(w)}^{\bar{\pi}}(w, 0) \geq \gamma \cdot \left(\frac{\bar{\pi}(v)}{2} - \frac{\bar{\pi}(u_1) + \bar{\pi}(u_2)}{2} - O(\gamma) \right) \geq \frac{\gamma\epsilon}{2} - O(\gamma^2) \gg \epsilon'.$$

Thus, by (10), we have $\bar{\pi}_{\mathsf{cr}(w)}(0|w) = 0$, i.e., $\bar{\pi}(w) = 1$. Since $Q_{\mathsf{cr}(v)}^{\bar{\pi}}(v, 1) = 0$, we have

$$Q_{\mathsf{cr}(v)}^{\bar{\pi}}(v, 0) - Q_{\mathsf{cr}(v)}^{\bar{\pi}}(v, 1) \geq \frac{\gamma}{2} - O(\gamma^2) \gg \epsilon'.$$

But then $\bar{\pi}_{\mathsf{cr}(v)}(1|v) = 0$, i.e., $\bar{\pi}(v) = 0$, which is a contradiction.

Similar reasoning applies in the case that $\bar{\pi}(v) < \bar{\pi}(u_1) + \bar{\pi}(u_2) - \epsilon$. $\square$

Note that technically, to prove Lemma 5.4, we needed to replace π by $\bar{\pi}$: thus the ϵ-approximate solution to $(V, \mathcal{G})$ produced in the proof of Theorem 5.3 is actually $\bar{\pi}$, which is fine since $\bar{\pi}$ is efficiently computable from π.

[8] Working through the computations, one may show that $\epsilon_0 = c \cdot \epsilon^{16}$ suffices, for some constant $c > 0$.

6 Simple stochastic games: are they hard?

In the previous section we showed that it is PPAD-hard to compute an ϵ_0-approximate Nash equilibrium in turn-based *general-sum* stochastic games, even when the discount factor γ is $1/2$. In this section we consider what seems to be an easier problem: what about *zero-sum* stochastic games? To keep things simple we consider the case that $A = 2$, so that the description length of a stochasitc game is polynomial in S, the number of states. A variant of value iteration easily establishes the following fact:

Proposition 6.1 ([Sha53])**.** *In a zero-sum turn-based stochastic game with S states and discount factor γ, an ϵ-approximate Nash equilibrium can be computed in time* $\mathrm{poly}(S, 1/(1-\gamma), \log(1/\epsilon))$.

One may contrast the result of Proposition 6.1 with the single-player setting (i.e., discounted MDPs), in which an ϵ-approximate Nash equilibrium can be computed in time $\mathrm{poly}(S, \log(1/(1-\gamma)), \log(1/\epsilon))$, using linear programming. One may wonder whether an analogous guarantee holds for two-player zero-sum turn-based stochastic games – concretely, we ask:

Problem 6.1 ([Con92])**.** Is there an algorithm that, given as input a 2-player zero-sum turn-based stochastic game with $\gamma = 1 - 2^{-S}$, runs in $\mathrm{poly}(S, 1/\epsilon)$ time, and outputs an ϵ-approximate Nash equilibrium?

Problem 6.1 was originally introduced in [Con92], where it was phrased in the slightly different (though equivalent) terminology of *simple stochastic games (SSGs)*. As such, we will refer (with slight abuse of terminology) to Problem 6.1 as the problem of computing Nash equilibria in SSGs. This is a major open problem, with various applications deriving from areas including complexity theory, logic, and algorithms:

- The decision-problem variant of Problem 6.1, namely deciding whether the optimal value of an SSG is larger than $1/2$, is in $\mathsf{NP} \cap \mathsf{coNP}$. Thus, it is very unlikely to be NP-hard.
- This decision-problem variant is logspace complete for the class of languages accepted by logspace-bounded randomized alternating Turing machines [Con92].
- Several other problems not known to have polynomial-time algorithms, such as *mean-payoff games* and *parity games*, are known to be reducible to SSGs. Moreover, parity games have applications in logic relating to μ-calculus and tree automata.
- In many ways, the problem of deciding SSGs can be seen as a generalization of linear programming. In fact, the best known algorithm for solving SSGs is a variant of the simplex algorithm with a randomized pivot rule. It runs in time $2^{O(\sqrt{n})}$ [Lud95]. There has been essentially no progress on improving this runtime for the last ~ 3 decades!

References

[Con92] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[DGP06] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, page 71–78, New York, NY, USA, 2006. Association for Computing Machinery.

[DGZ23] Constantinos Daskalakis, Noah Golowich, and Kaiqing Zhang. The complexity of markov equilibrium in stochastic games. In Gergely Neu and Lorenzo Rosasco, editors, *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 4180–4234. PMLR, 12–15 Jul 2023.

[FGK23] Dylan J Foster, Noah Golowich, and Sham M. Kakade. Hardness of independent learning and sparse equilibrium computation in Markov games. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10188–10221. PMLR, 23–29 Jul 2023.

[JLWY21] Chi Jin, Qinghua Liu, Yuanhao Wang, and Tiancheng Yu. V-learning – a simple, efficient, decentralized algorithm for multiagent rl, 2021.

[Lud95] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.

[Rub18] Aviad Rubinstein. Inapproximability of nash equilibrium. *SIAM Journal on Computing*, 47(3):917–959, 2018.

[Sha53] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.