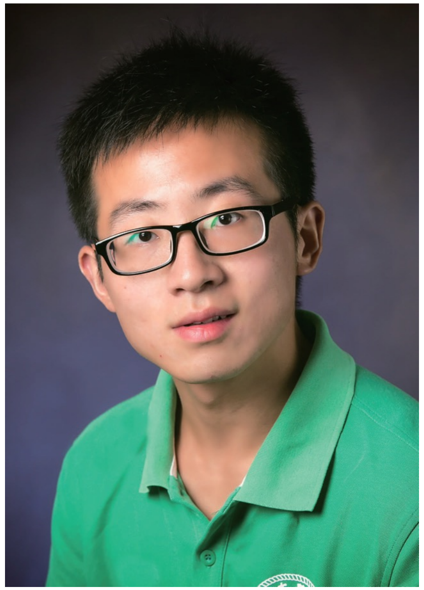
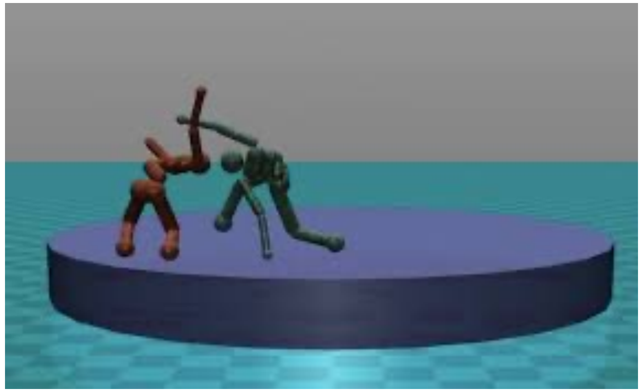
The Complexity of Markov Equilibrium in Stochastic Games

Constantinos Daskalakis Noah Golowich Kaiqing Zhang



Reinforcement learning applications

- Involve multiple players!



- **This talk**: investigate some basic questions regarding equilibrium *computation* and *learning* in multi-player tabular RL environments (stochastic games)

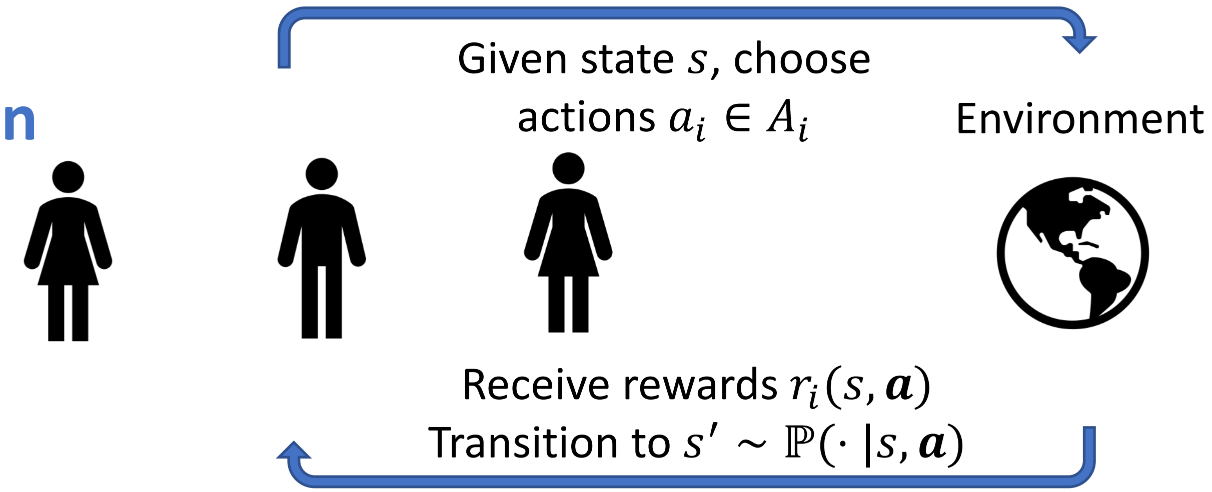
Note: concurrent work by [Jin-Muthukumar-Sidford, ‘22] which proves some similar results

Main results: summary

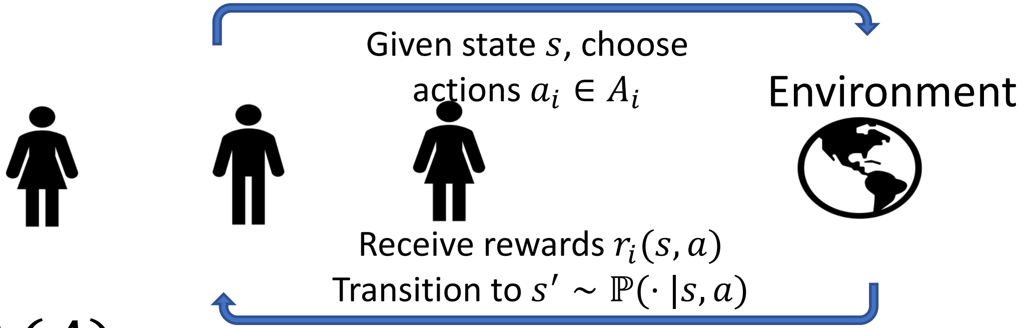
1. Hardness result for computing **stationary** CCE in stochastic games
2. Decentralized algorithm for learning **nonstationary** CCE efficiently

Stochastic games: preliminaries

- Infinite-horizon discounted m-player stochastic game $G = (S, A, \mathbb{P}, r, \gamma, \mu)$:
  - S is a finite set of **states**
  - $A = A_1 \times \cdots \times A_m$ is a **joint action set** (agent $i \in [m]$ has **action set** A_i)
    - Denote joint actions in boldface, i.e., $\boldsymbol{a} = (a_1, \dots, a_m) \in A$
  - $\mathbb{P}(s'|s, \boldsymbol{a})$, for $s, s' \in S$, $\boldsymbol{a} \in A$, gives **transition kernel**
  - $r = (r_1, \dots, r_m)$ is tuple of **reward functions**, where $r_i(s, \boldsymbol{a})$ gives **reward function** of agent i
  - $\gamma \in (0,1)$ is **discount factor**
  - $\mu \in \Delta(S)$ is **initial state distribution**



Nash equilbrium



- **(Markov) stationary policy**: mapping $\pi: S \to \Delta(A)$
- **Value function** for player i: (below $\boldsymbol{a}_h = (a_{h1}, \dots, a_{hm})$)

$$V_i^\pi(s) \coloneqq (1-\gamma) \cdot \mathbb{E}_{(s_1, \boldsymbol{a}_1, s_2, \boldsymbol{a}_2, \dots) \sim (\mathbb{P}, \pi)} \left[\sum_{h=1}^{\infty} \gamma^{h-1} \cdot r_i(s_h, \boldsymbol{a}_h) \mid s_1 = s \right]$$

- Also define: $V_i^\pi(\mu) \coloneqq \mathbb{E}_{s\sim\mu}[V_i^\pi(s)]$

Product policy: $\pi(s) \in \Delta(A_1) \times \cdots \times \Delta(A_m)$ is a *product distribution* for all s

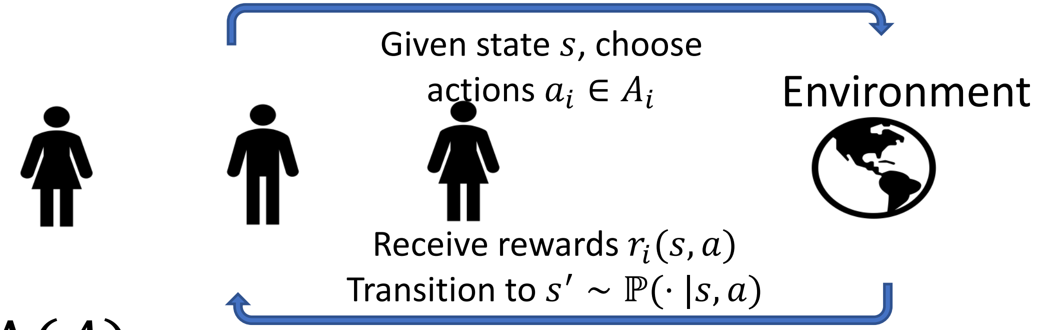
Definition: For $\epsilon > 0$:

- **ϵ-apx stationary Nash equilibrium** is a stationary **product** policy π so that for all i,

$$\max_{\pi_i'} V_i^{\pi_i', \pi_{-i}}(\mu) - V_i^\pi(\mu) \le \epsilon.$$

Problem: for a game with a single state, ϵ-stationary Nash is just ϵ-Nash in a normal-form game, which is PPAD-complete!

Coarse correlated equilibrium



- **(Markov) stationary policy**: mapping $\pi : S \to \Delta(A)$
- **Value function** for player i: (below $\boldsymbol{a}_h = (a_{h1}, \dots, a_{hm})$)

$$V_i^\pi(s) := (1-\gamma) \cdot \mathbb{E}_{(s_1, \boldsymbol{a}_1, s_2, \boldsymbol{a}_2, \dots) \sim (\mathbb{P}, \pi)} \left[\sum_{h=1}^{\infty} \gamma^{h-1} \cdot r_i(s_h, \boldsymbol{a}_h) \mid s_1 = s \right]$$

- Also define: $V_i^\pi(\mu) := \mathbb{E}_{s \sim \mu}[V_i^\pi(s)]$

Perhaps: expect Coarse correlated equilibrium (CCE) to be tractable here (as it is in normal form games)

Definition: For $\epsilon > 0$:

- **ϵ-apx stationary CCE** is a stationary policy π so that for all players i,

$$\max_{\pi_i'} V_i^{\pi_i', \pi_{-i}}(\mu) - V_i^\pi(\mu) \le \epsilon.$$

- **ϵ-apx perfect stationary CCE** is a stationary policy π so that for all players i **and all s**,

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

Background: complexity class PPAD

- **Total** search problems: solution (e.g., stationary CCE) always exists
- PPAD ("**P**olynomial **P**arity **A**rguments on **D**irected Graphs"):
  - Roughly speaking: class consisting of total search problems which have a polynomial-time reduction to the **End-Of-The-Line problem**
- **End-Of-The-Line (EOTL) problem**: given a directed graph G with exponentially many vertices where each vertex has at most one predecessor and successor, together with a source: find a sink!

  - G is specified succinctly by having a (poly-size) circuit return predecessor + successor of each vertex
  - **PPAD-hard** problems: as hard as EOTL (likely needs super-polynomial time)

PPAD-hardness of stationary CCE

- ϵ-**stationary CCE** is policy π so that for all i,
$$\max_{\pi_i'} V_i^{\pi_i',\pi_{-i}}(\mu) - V_i^{\pi}(\mu) \le \epsilon.$$
- ϵ-**perfect stationary CCE** is policy π so that for all i **and all s**,
$$\max_{\pi_i'} V_i^{\pi_i',\pi_{-i}}(s) - V_i^{\pi}(s) \le \epsilon.$$

Theorem [Daskalakis-G-Zhang, '22]: For some **constant $\epsilon > 0$**, computing **ϵ-perfect** stationary CCE in **2-player** stochastic games with discount factor $\gamma = 1/2$ is PPAD-hard.

Theorem [Daskalakis-G-Zhang, '22]: For some **constant $\epsilon > 0$**, computing ϵ-stationary CCE in **2-player** stochastic games with discount factor $\gamma = 1/2$ is PPAD-hard under the "**PCP for PPAD conjecture**".

Theorem [Daskalakis-G-Zhang, '22]: For some constant $c > 0$, computing $\boldsymbol{c/n}$-stationary CCE in **2-player, $\boldsymbol{n}$**-state stochastic games with discount factor $\gamma = 1/2$ is PPAD-hard.

- Larger γ always harder: so get PPAD-hardness for all $\gamma \geq \frac{1}{2}$
- **Known** [Deng-Li-Mguni-Wang-Yang, '21],[Jin-Muthukumar-Sidford, '22]: computing (perfect) stationary CCE is in PPAD, so all problems above are PPAD-complete.

Proof overview of hardness result

Concurrent work *[Jin-Muthukumar-Sidford, '22]*: get below theorem for $|S|$-player games (i.e., weaker result)

First step: consider **turn-based stochastic games**: special case where 1 player acts at each state

- Key point: CCE and Nash equilibria are equivalent in turn-based games

Theorem [Daskalakis-G-Zhang, '22]: For some **constant** $\epsilon > 0$, computing ϵ-**perfect** stationary Nash equilibrium in **2-player turn-based** stochastic games with discount factor $\gamma = 1/2$ is PPAD-hard.

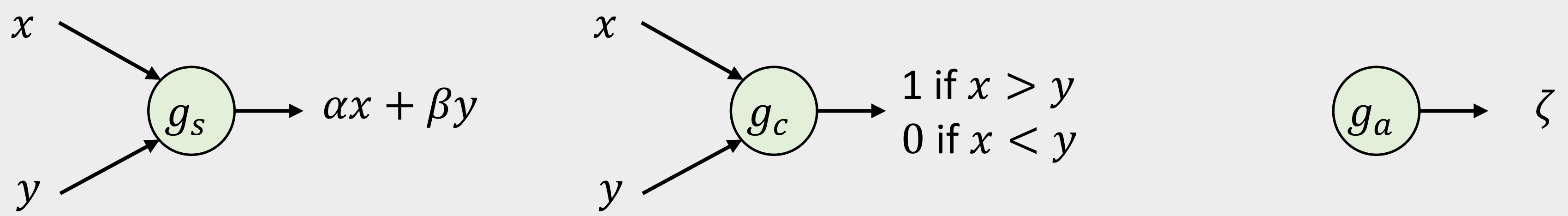
Proof idea: Reduce from the (PPAD-hard) **ϵ-generalized circuit (GCircuit)** problem:

Definition (ϵ-Generalized circuit problem; informal): Given a circuit, i.e., collection of gates G, each gate being one of the following:

(Weighted) ***summation*** *gate* g_s *(parametrized by* $\alpha, \beta \in \mathbb{R}$*)*

Comparison *gate* g_c

Assignment *gate* g_a *(parametrized by* $\zeta \in \{0,1\}$*)*



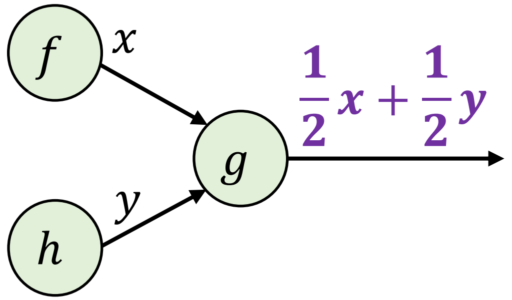
x, y are outputs from other gates

Problem: find an assignment of real values to all wires of the circuit such that constraints of all gates are satisfied up to $\pm\epsilon$

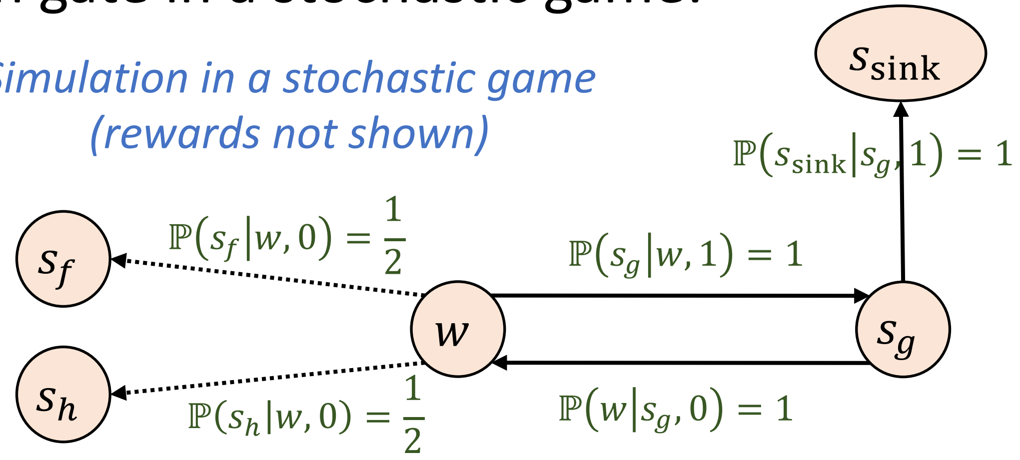
Proof overview of hardness result: $|S|$-player games

- Known: Finding assignment to an ϵ-GCircuit instance is PPAD-hard [Daskalakis-Golberg-Papadimitriou, '06],[Chen-Deng-Teng,'09],[Rubinstein, '18]:
- Our proof: shows how to "simulate" each gate in a generalized circuit using $O(1)$ states in a turn-based stochastic game where each state has 2 actions (i.e., $A_i = \{0,1\}$)
- First establish the (easier) result where each state in the game is controlled by a different player (as in *[Jin-Muthukumar-Sidford, '22];* uses ideas from *[Daskalakis-Goldberg-Papadimitriou, '06]*)
- Example: implement summation gate in a stochastic game:

Summation gate g
(suppose $\alpha = \beta = \frac{1}{2}$ for simplicity):



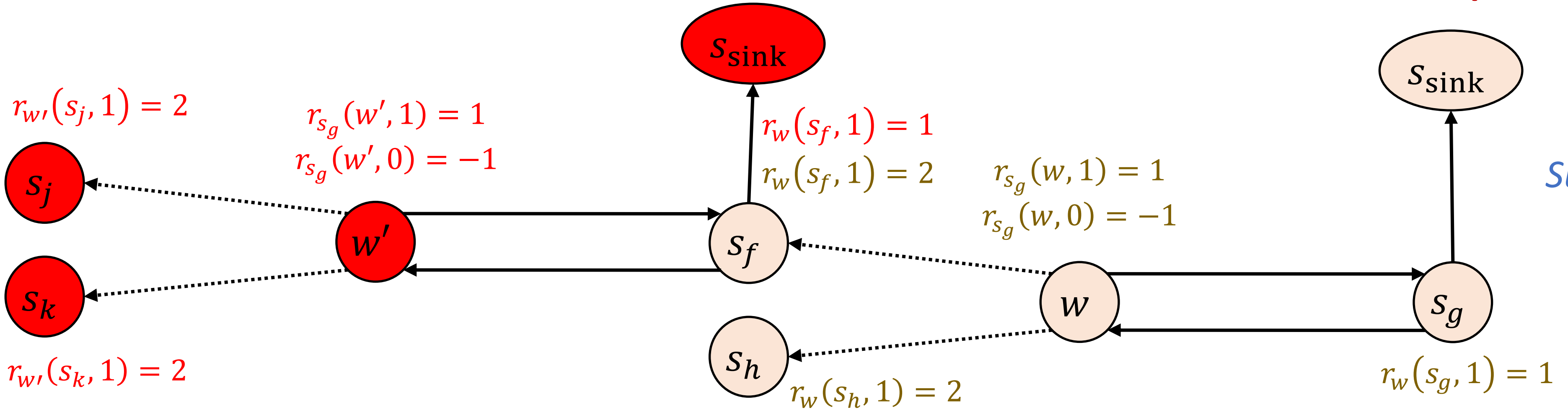
Simulation in a stochastic game
(rewards not shown)



Observation: since game is turn-based and $A_i = \{0,1\}$, stationary policy $\pi: S \to \Delta(A)$ is simply a mapping $\pi: S \to [0,1]$

Lemma [ours; informal]: For any $\epsilon > 0$, exists $\epsilon' > 0$ so that for any ϵ'-stationary NE $\pi: S \to [0,1]$, it holds that $\boldsymbol{\pi(s_g) = \frac{1}{2}\pi(s_f) + \frac{1}{2}\pi(s_h) \pm \epsilon}$.

Proof overview of hardness result: 2-player games



Summation gate with $\alpha = \beta = 1$; all rewards not shown are 0

- Issue when trying to prove hardness for 2-player games: rewards from different gadgets may *conflict* with one another!
- Example of conflict: try to assign all helper nodes "w" to one player, all non-helper nodes to the other player
  - Requirement that $r_w(s_f, 1) = 2$ above may conflict with requirement that (e.g.) $r_w(s_f, 1) = 1 \neq 2$ from some other weighted summation gate
- Solution: show how to "pre-process" any generalized circuit instance (using a notion of "*valid coloring*" we introduce) to avoid conflicts
  - Pre-processing uses the unary-to-binary and binary-to-unary constructions in *[Rubinstein, '18]*

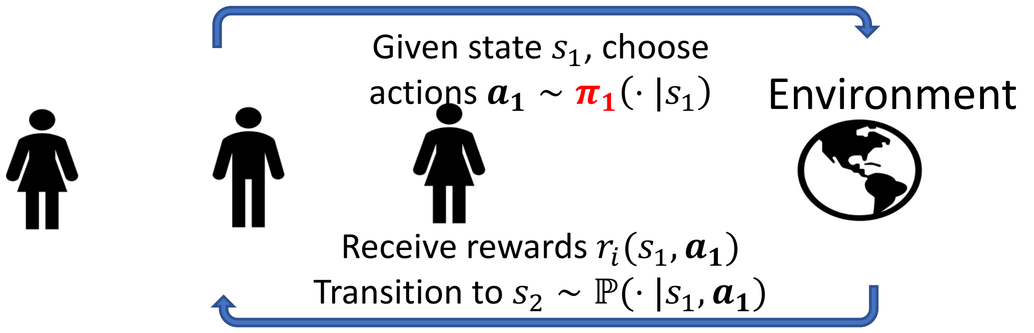
What is computable?

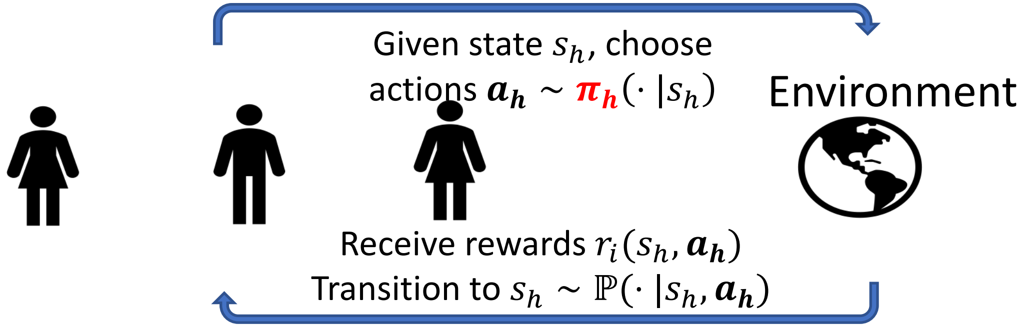
- How to get around PPAD-hardness: allow for **nonstationary CCE**:
- **Nonstationary policy** π is a collection $\pi = (\pi_1, \pi_2, \ldots)$, where each $\pi_h : S \to \Delta(A)$
  - Allow choice of actions to depend on the time step

Definition: For $\epsilon > 0$, an **ϵ-nonstationary coarse correlated equilibrium (CCE)** is a **nonstationary policy** π so that for all players i,

$$\max_{\pi'_i} V_i^{\pi'_i, \pi_{-i}}(\mu) - V_i^{\pi}(\mu) \leq \epsilon.$$

Nonstationary CCE same as stationary CCE, except policy no longer stationary



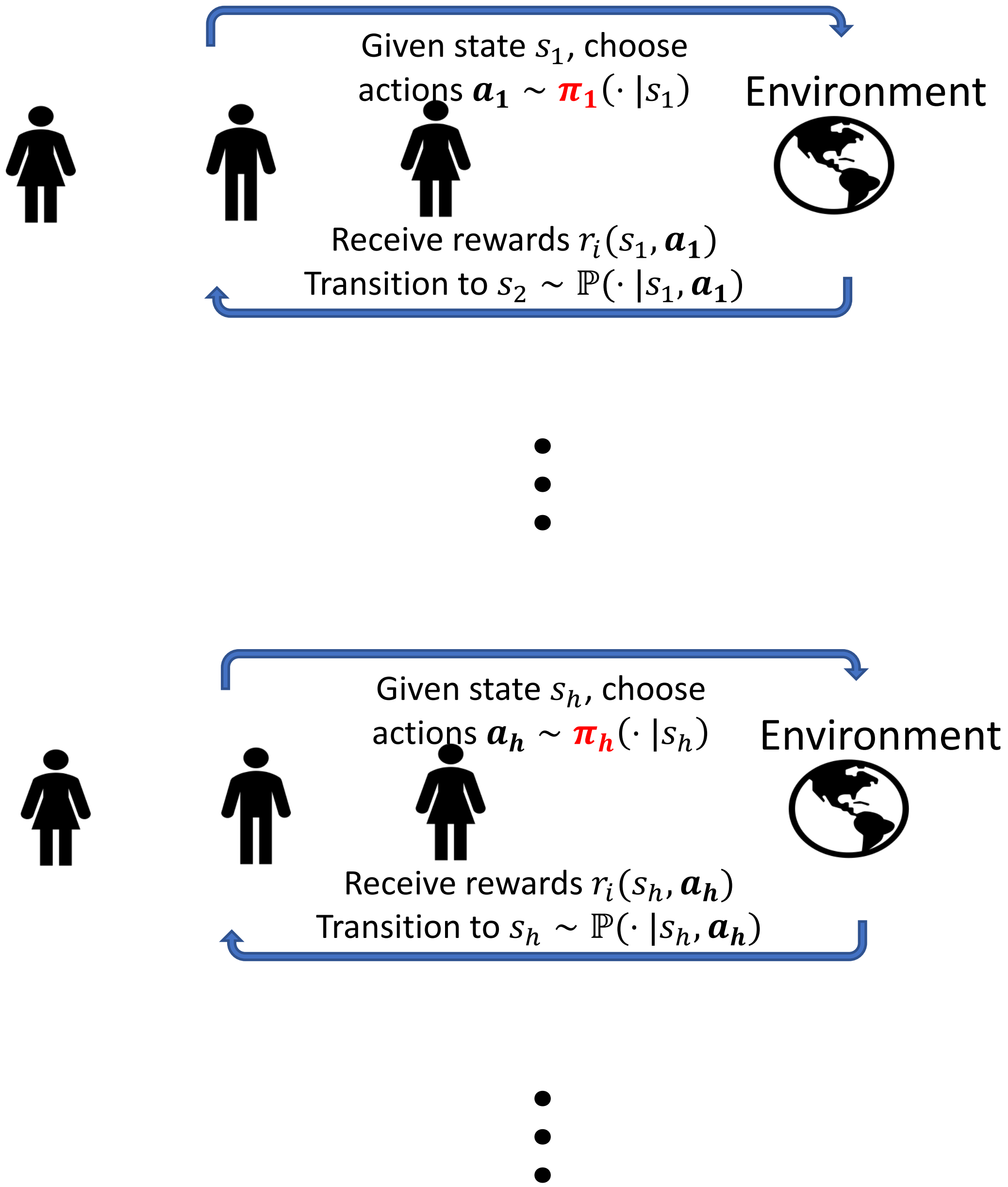


What is computable?

Definition: For $\epsilon > 0$, an ***ϵ*-nonstationary coarse correlated equilibrium (CCE)** is a **nonstationary policy** π so that for all players i,

$$\max_{\pi_i'} V_i^{\pi_i', \pi_{-i}}(\mu) - V_i^{\pi}(\mu) \leq \epsilon.$$

- **Fact (folklore):** ϵ-nonstationary CCE may be computed in poly time if stochastic game is **known**
  - How? Simply use backwards induction and truncate after $\frac{\log 1/\epsilon}{1-\gamma}$ steps (more detail later)



Can we **learn** nonstationary CCE (i.e., if the stochastic game is unknown)?

Prior work -- 2 groups of work:

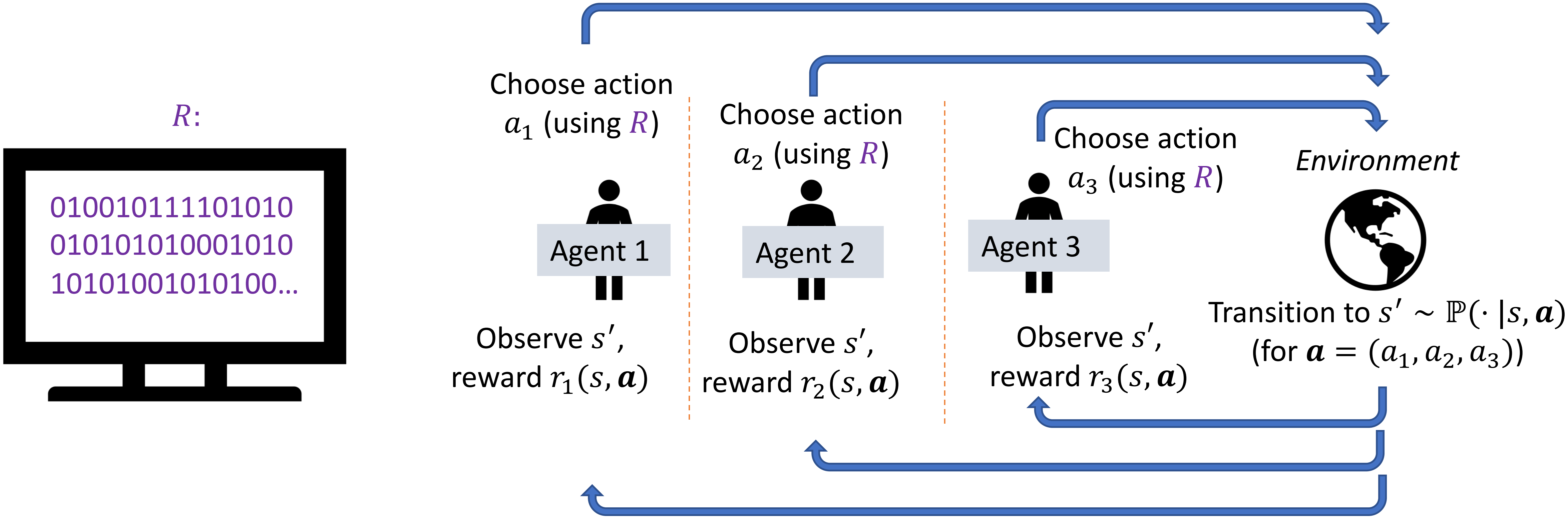
1. Requires *exponential time* in number of players *[Liu-Yu-Bai-Jin,'21]*
   - **Curse of multi-agents**
   - Algorithm is model-based (learn entire transitions $\mathbb{P}(\cdot|s, \boldsymbol{a})$)
   - Does output Markov policy
2. Is poly-time, but *does not learn a Markov policy [Song-Mei-Bai,'21],[Jin-Liu-Wang-Liu,'21],[Mao-Basar,'21']*
   - V-learning algorithm – avoids curse of multi-agents using regret minimization algorithm at each state
   - Policies output by V-learning are *history-dependent*: is complicated function of the policies played by bandit learners in the course of the algorithm
   - Is decentralized…

Additional desideratum: **decentralized** algorithms

Decentralized model:

Not needed in V-learning

- Agents only see states, their own actions, and their own rewards
- Agents have access to *common randomness* R (used to correlate their actions during course of algorithm)
- No communication between agents allowed



Guarantee for decentralized learning

Learning setup (**Episodic PAC-RL model**):

- **"Episodic"**: Only access to game is ability to repeatedly sample *trajectories* (i.e., sequence of $s_h, \boldsymbol{a}_h$) in *decentralized setting*
- **"PAC RL":** At end of interaction, output ϵ-CCE whp
- **Technical point**: Need to be able to *truncate* trajectories – we assume that trajectories are truncated at $H = \frac{\log \frac{1}{\epsilon}}{1-\gamma}$ steps
  - Our result holds in finite-horizon setting too (no need to truncate)

Environment

Episode 1: each player i chooses policy & sees
$s_1, a_{i1}, r_i(s_1, \boldsymbol{a}_1), s_2, a_{i2}, r_i(s_2, \boldsymbol{a}_2), \ldots$

Episode 2: each player i chooses policy & sees
$s_1, a_{i1}, r_i(s_1, \boldsymbol{a}_1), s_2, a_{i2}, r_i(s_2, \boldsymbol{a}_2), \ldots$

Episode 3: each player i chooses policy & sees
$s_1, a_{i1}, r_i(s_1, \boldsymbol{a}_1), s_2, a_{i2}, r_i(s_2, \boldsymbol{a}_2), \ldots$

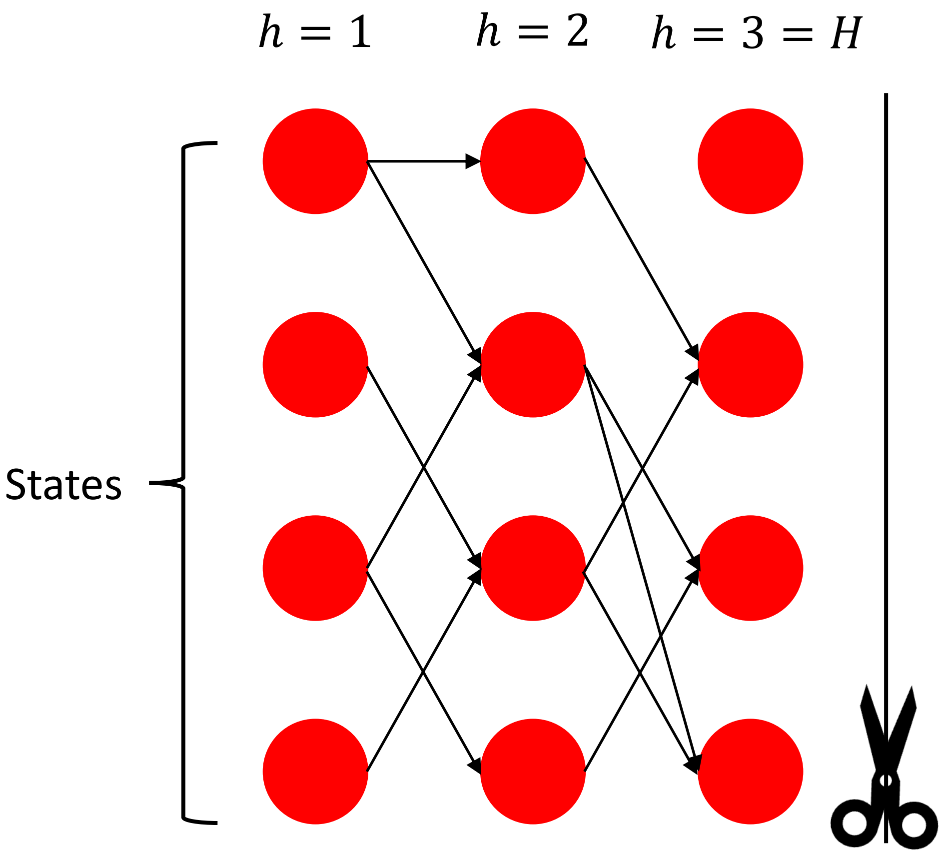
$\vdots$

Theorem [DGZ, '22]: There is a *decentralized* learning algorithm (SPoCMAR) that requires $\tilde{O}\left(\frac{S^3 \cdot \max_{i\in[m]}\{A_i\}}{\epsilon^3 \cdot (1-\gamma)^{10}}\right)$ samples, polynomial time, and outputs an ϵ-nonstationary **Markov** CCE (whp).

Warm-up: computing CCE if game is known

Fact (folklore): ϵ-nonstationary CCE may be computed in poly time if stochastic game is **known**

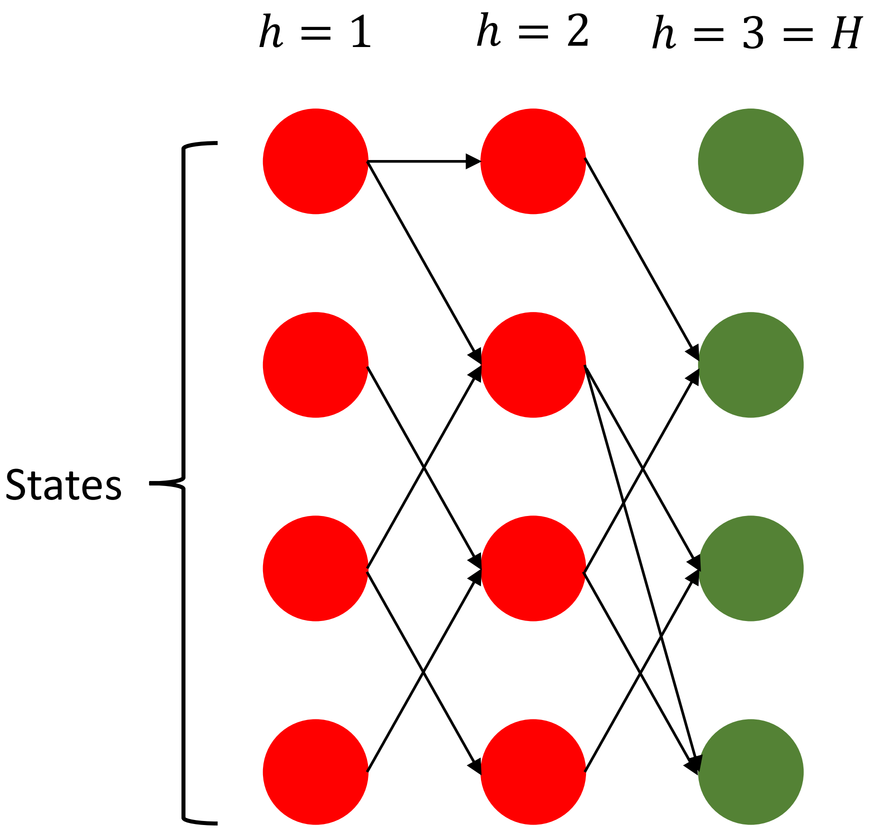
- Ignore all steps after $H := \frac{\log 1/\epsilon}{1-\gamma}$ steps *(safe since they contribute $< \epsilon$ to value functions)*



Warm-up: computing CCE if game is known

Fact (folklore): ϵ-nonstationary CCE may be computed in poly time if stochastic game is **known**

- Ignore all steps after $H := \frac{\log 1/\epsilon}{1-\gamma}$ steps *(safe since they contribute $< \epsilon$ to value functions)*



- Construct functions $V_{i,h}: S \to \mathbb{R}$, policy $\pi_h: S \to \Delta(A)$ for $h = H + 1, H, H - 1, \ldots, 1$ inductively:

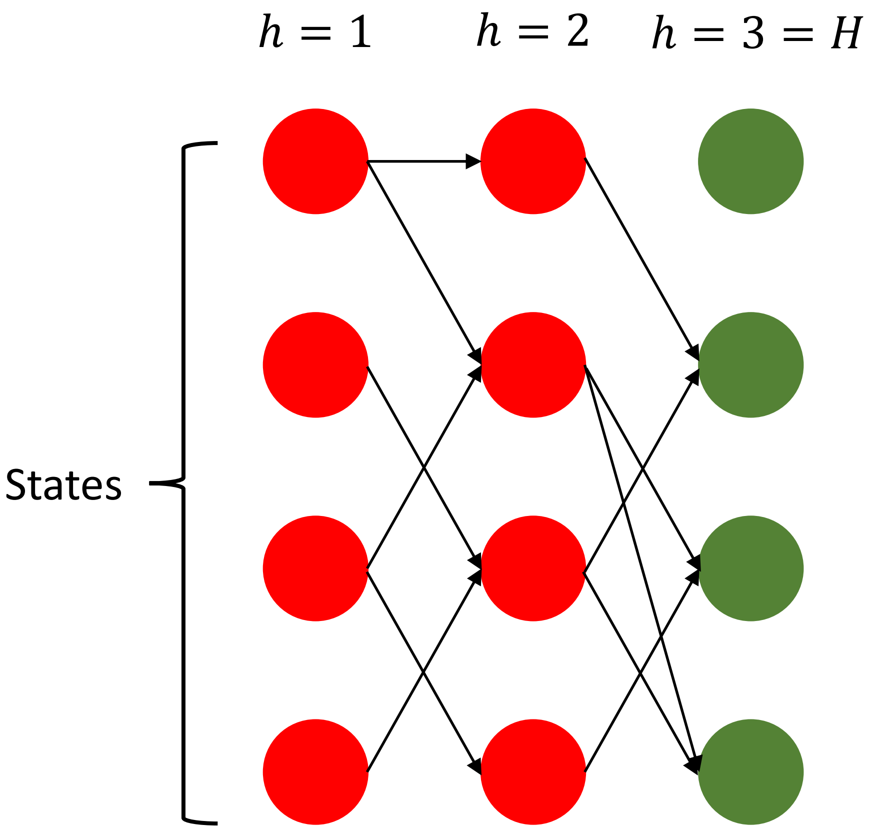
Base Case:

$V_{i,H+1}(s) \leftarrow 0$ for all s, i

Warm-up: computing CCE if game is known

Fact (folklore): ϵ-nonstationary CCE may be computed in poly time if stochastic game is **known**

- Ignore all steps after $H := \frac{\log 1/\epsilon}{1-\gamma}$ steps *(safe since they contribute $< \epsilon$ to value functions)*



- Construct functions $V_{i,h}: S \to \mathbb{R}$, policy $\pi_h: S \to \Delta(A)$ for $h = H + 1, H, H - 1, \ldots, 1$ inductively:

Base Case:

$V_{i,H+1}(s) \leftarrow 0$ for all s, i

Inductive step:

1. Assume given $V_{i,h+1}: S \to \mathbb{R}$ (e.g., $h = 2$)
2. For each $s \in S, i \in [m]$, construct mapping $F_{is}: A \to \mathbb{R}$:

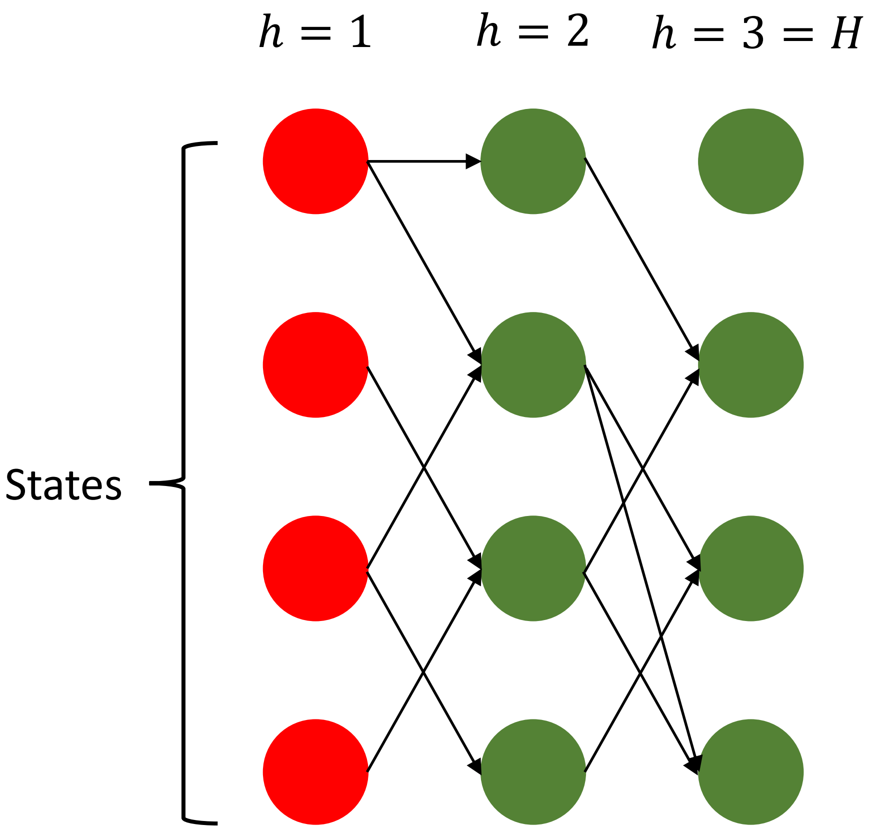
$$F_{is}(\boldsymbol{a}) := r_i(s, \boldsymbol{a}) + \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,\boldsymbol{a})}[V_{i,h+1}(s')]$$

3. Compute a ϵ-CCE of each $(F_{1s}, \ldots, F_{ms})$, and let that be $\pi_h(s) \in \Delta(A)$

Warm-up: computing CCE if game is known

Fact (folklore): ϵ-nonstationary CCE may be computed in poly time if stochastic game is **known**

- Ignore all steps after $H := \frac{\log 1/\epsilon}{1-\gamma}$ steps *(safe since they contribute $< \epsilon$ to value functions)*



- Construct functions $V_{i,h}: S \to \mathbb{R}$, policy $\pi_h: S \to \Delta(A)$ for $h = H+1, H, H-1, \ldots, 1$ inductively:

Base Case:

$V_{i,H+1}(s) \leftarrow 0$ for all s, i

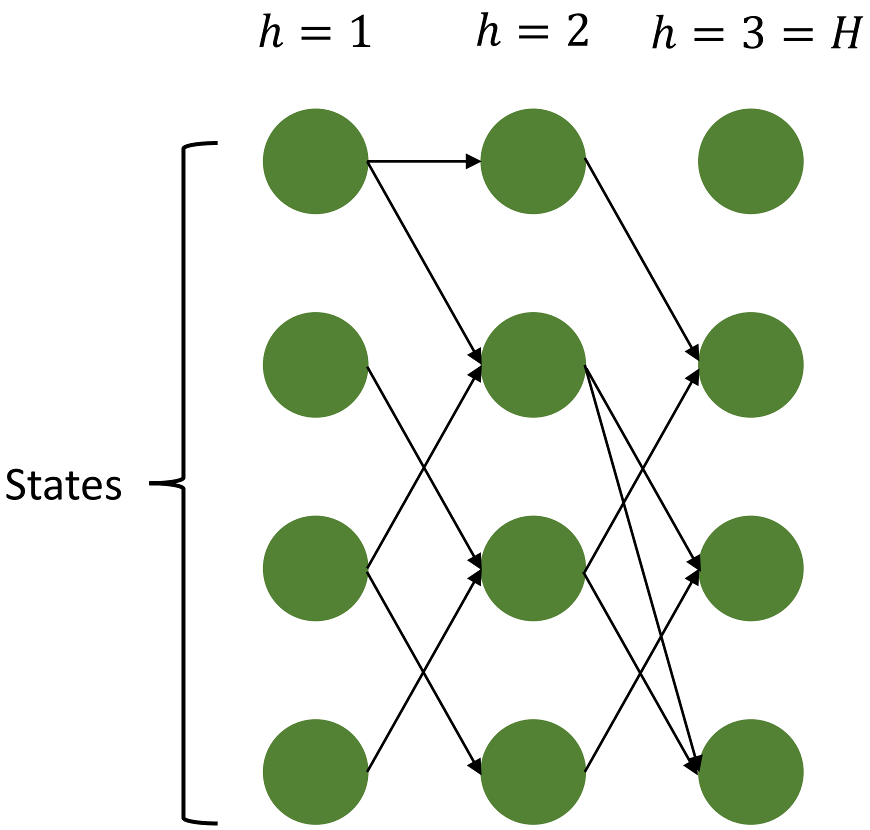
Inductive step:

1. Assume given $V_{i,h+1}: S \to \mathbb{R}$ (e.g., $h = 2$)
2. For each $s \in S, i \in [m]$, construct mapping $F_{is}: A \to \mathbb{R}$:
   $$F_{is}(\boldsymbol{a}) := r_i(s, \boldsymbol{a}) + \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,\boldsymbol{a})}[V_{i,h+1}(s')]$$
3. Compute a ϵ-CCE of each $(F_{1s}, \ldots, F_{ms})$, and let that be $\pi_h(s) \in \Delta(A)$
4. Let $V_{i,h}(s) := \mathbb{E}_{\boldsymbol{a} \sim \pi_h(s)}[F_{is}(\boldsymbol{a})]$

Warm-up: computing CCE if game is known

Fact (folklore): ϵ-nonstationary CCE may be computed in poly time if stochastic game is **known**

- Ignore all steps after $H := \frac{\log 1/\epsilon}{1-\gamma}$ steps *(safe since they contribute $< \epsilon$ to value functions)*



- Construct functions $V_{i,h}: S \to \mathbb{R}$, policy $\pi_h: S \to \Delta(A)$ for $h = H+1, H, H-1, \dots, 1$ inductively:

Base Case:
$V_{i,H+1}(s) \leftarrow 0$ for all s, i

Inductive step:

1. Assume given $V_{i,h+1}: S \to \mathbb{R}$ (e.g., $h = 1$)
2. For each $s \in S, i \in [m]$, construct mapping $F_{is}: A \to \mathbb{R}$:
$$F_{is}(\boldsymbol{a}) := r_i(s, \boldsymbol{a}) + \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,\boldsymbol{a})}[V_{i,h+1}(s')]$$
3. Compute a ϵ-CCE of each $(F_{1s}, \dots, F_{ms})$, and let that be $\pi_h(s) \in \Delta(A)$
4. Let $V_{i,h}(s) := \mathbb{E}_{\boldsymbol{a} \sim \pi_h(s)}[F_{is}(\boldsymbol{a})]$

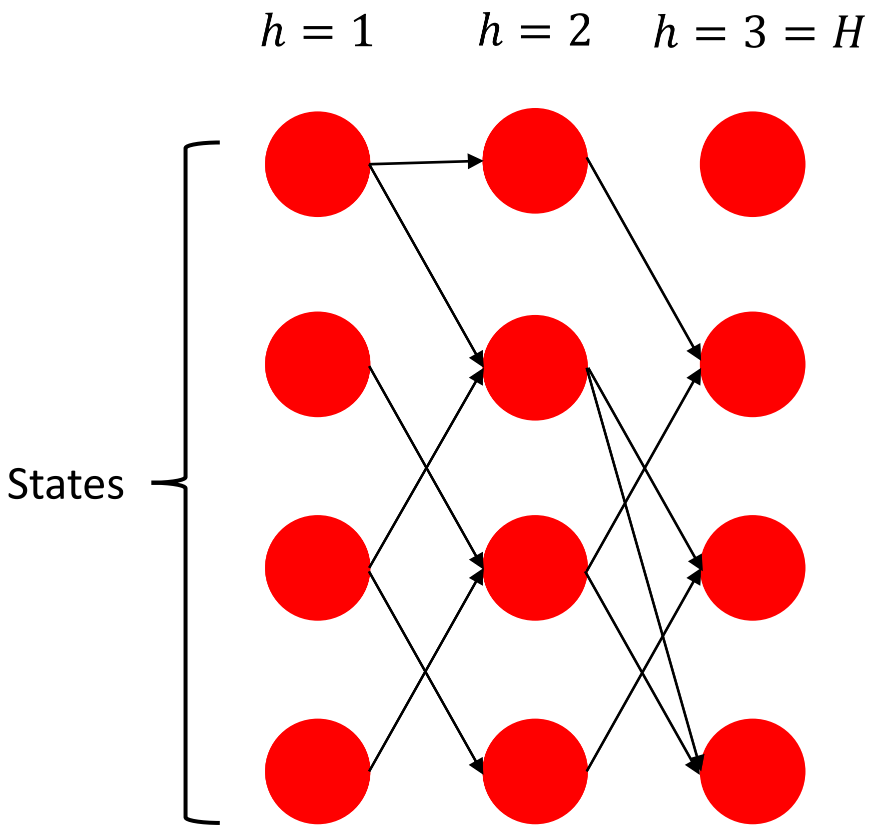
Output: $\boldsymbol{\pi} := (\boldsymbol{\pi_1}, \dots, \boldsymbol{\pi_H})$

Actually: use bandit no-regret learner here

SPoCMAR: Stage-Based Policy Cover for Multi-Agent Learning with Rmax

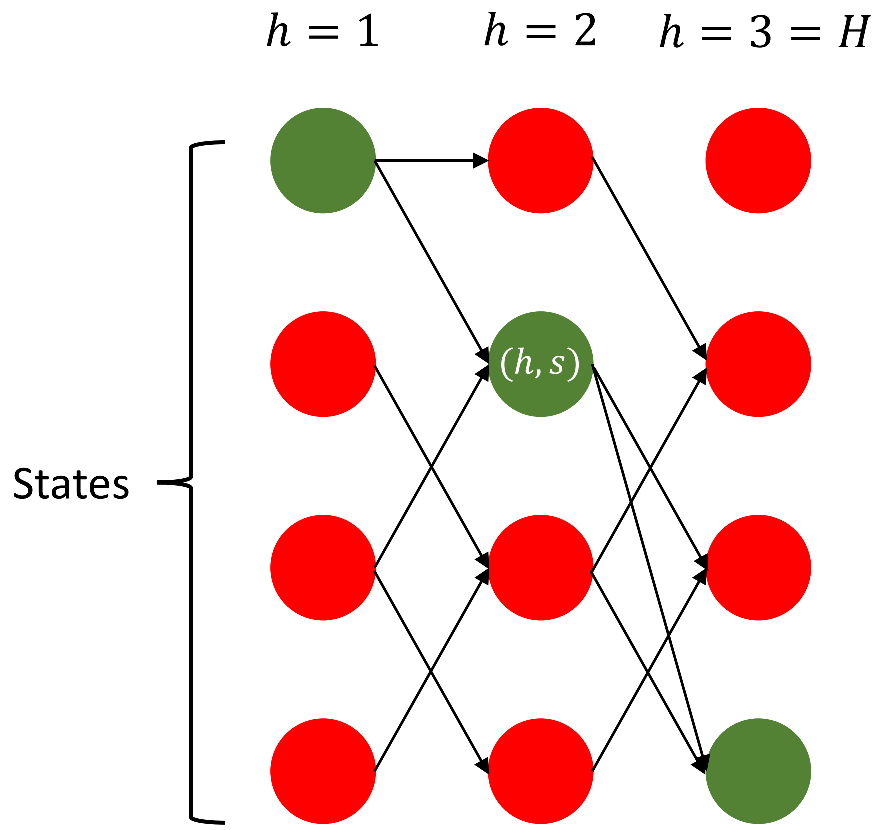
- What to do if the game is not known?
  - Idea in **V-learning**: replace computation of CCE of games F_{is} with *no-regret learner*, update $V_{i,h}$ *incrementally*
  - Issue: due to interdependence between V-updates in V-learning, don't get Markov policy
- Our solution: use a combination of **multi-stage** algorithm and **policy cover** to allow us to compute a Markov (nonstationary) policy
  - These tools make it tricky to use UCB bonuses (as in V-learning)
  - So instead we use **Rmax**-type bonuses *[Brafman-Tennenholz, '02]*, which leads to $\frac{1}{\epsilon^3}$ sample complexity (as opposed to the tight $\frac{1}{\epsilon^2}$)

SPoCMAR: Stage-Based Policy Cover for Multi-Agent Learning with Rmax



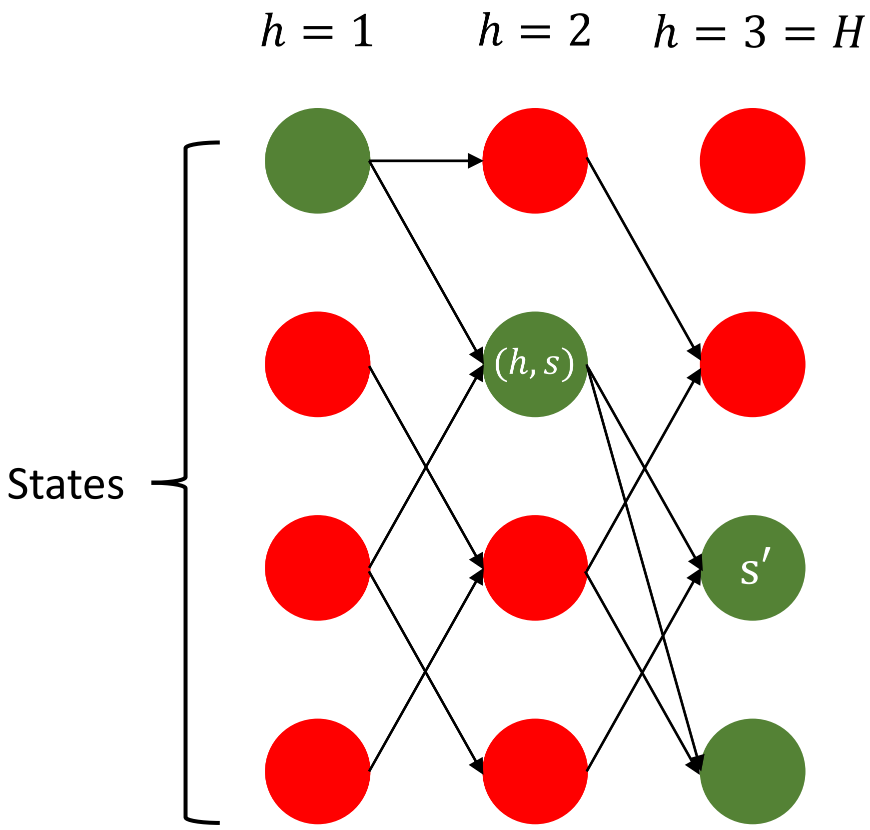
- Initially: all states unvisited, $V_{i,H+1}(s) = 0$ for all s

SPoCMAR: Stage-Based Policy Cover for Multi-Agent Learning with Rmax



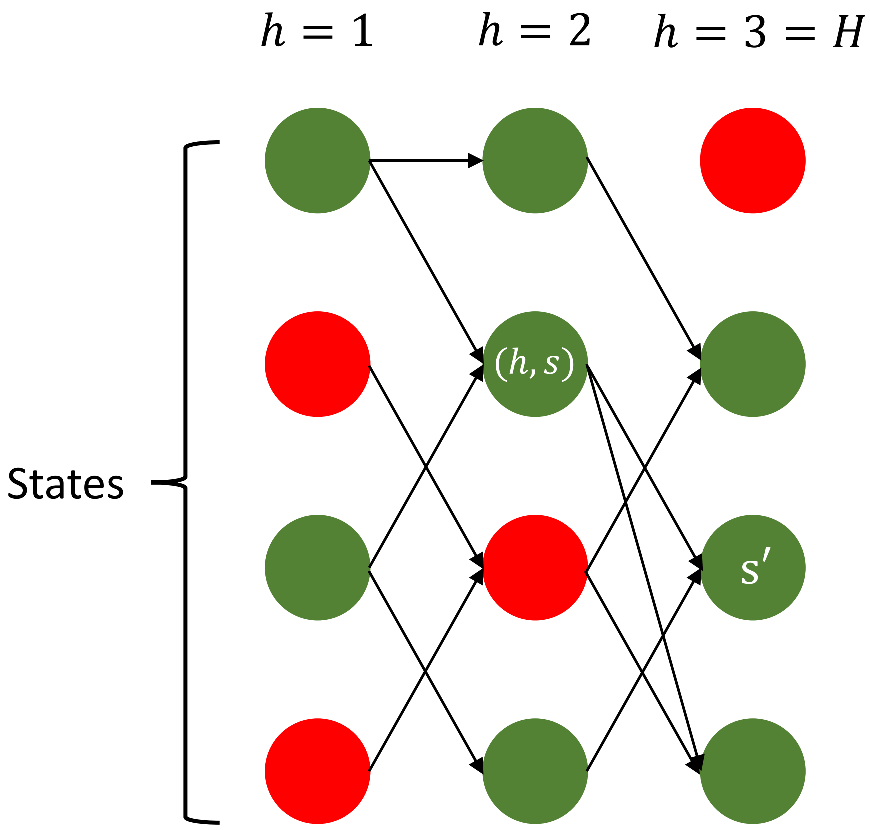
- Initially: all states unvisited, $V_{i,H+1}(s) = 0$ for all s
- At each **stage**: some subset W of pairs (h, s) are "known", i.e., exists policy π^{hs} that visits (h, s) with nontrivial probability
  - Set of π^{hs} known as **policy cover**
- At each stage: for all $(h, s) \in W$, play π^{hs} so as to reach (h, s), then play *bandit no-regret learner* at (h, s), transition to s'
  - Reward for bandit learner: $V_{i,h+1}(s')$ function, computed inductively
- At end of stage: average rewards from bandit learner to compute $V_{ih}(s)$ for all $(h, s) \in W$
- What about $(h, s) \notin W$? $V_{ih}(s) \coloneqq H + 1 - h$ (**Rmax bonuses**)

SPoCMAR: Stage-Based Policy Cover for Multi-Agent Learning with Rmax



- Initially: all states unvisited, $V_{i,H+1}(s) = 0$ for all s
- At each **stage**: some subset W of pairs (h, s) are "known", i.e., exists policy π^{hs} that visits (h, s) with nontrivial probability
  - Set of π^{hs} known as **policy cover**
- At each stage: for all $(h, s) \in W$, play π^{hs} so as to reach (h, s), then play *bandit no-regret learner* at (h, s), transition to s'
  - Reward for bandit learner: $V_{i,h+1}(s')$ function, computed inductively
- At end of stage: average rewards from bandit learner to compute $V_{ih}(s)$ for all $(h, s) \in W$
- What about $(h, s) \notin W$? $V_{ih}(s) := H + 1 - h$ (**Rmax bonuses**)
- *If bandit learner at (h, s) visits "unknown" state s' at step $h + 1$:*
  - Can use it to compute a cover policy $\pi^{h+1,s'}$ (progress!)

SPoCMAR: Stage-Based Policy Cover for Multi-Agent Learning with Rmax



- Initially: all states unvisited, $V_{i,H+1}(s) = 0$ for all s
- At each **stage**: some subset W of pairs (h, s) are "known", i.e., exists policy π^{hs} that visits (h, s) with nontrivial probability
  - Set of π^{hs} known as **policy cover**
- At each stage: for all $(h, s) \in W$, play π^{hs} so as to reach (h, s), then play *bandit no-regret learner* at (h, s), transition to s'
  - Reward for bandit learner: $V_{i,h+1}(s')$ function, computed inductively
- At end of stage: average rewards from bandit learner to compute $V_{ih}(s)$ for all $(h, s) \in W$
- What about $(h, s) \notin W$? $V_{ih}(s) := H + 1 - h$ (**Rmax bonuses**)
- *If bandit learner at (h, s) visits "unknown" state s' at step $h + 1$:*
  - Can use it to compute a cover policy $\pi^{h+1,s'}$ (progress!)
- Otherwise: policies from bandit learners can be concatenated to produce **output policy,** ☺

Conclusions/Open problems

Thank you for listening!

Summary of computation costs for finding CCE in general-sum stochastic games:

	Markov-Stationary	Markov-Nonstationary	Non-Markov
Computation	**PPAD-hard [our paper]**	[folklore]: Polynomial	[folklore]: Polynomial
Learning	**PPAD-hard [our paper]**	[LYBJ,'21]: Exponential (in #players) **Polynomial [our paper]**	[SMB,'21],[MB,'21],[JLWY,'21]: Polynomial (via V-learning)

Open questions:

- Can we get PPAD-hardness of finding ϵ-stationary CCE (non-perfect) for constant ϵ without assuming "PCP for PPAD conjecture"?
- Tighter sample complexity for upper bound?
- More natural/simpler algorithm instead of SPoCMAR?
- Extend upper bound results to settings with (e.g., linear) function approximation?

SPoCMAR: Stage-Based Policy Cover for Multi-Agent Learning with Rmax

1. Maintain a set $\boldsymbol{\Pi}^{\mathbf{cover}} \coloneqq \{\pi^{hs} : h \in [H], s \in S\}$ denoting **policy cover** (initially $\pi^{hs} = \perp$ for all h, s)
2. Maintain a set $W \subset [H] \times S$ of "well-visited" states (initialized to $\emptyset$)
3. For each **stage** $q \geq 1$:
   - Initialize $V_{i,H+1}(s) \leftarrow 0$ for all agents i, states s
   - For $h = H, H-1, \dots, 1$:
     - A. Each player initializes an adversarial bandit no-regret learner at each state s
     - B. For each non-null policy $\boldsymbol{\pi} \in \boldsymbol{\Pi}^{\boldsymbol{cover}}$: choose actions according to π up to step $h-1$, then according to the bandit no-regret learners at step h
       - Sample a trajectory: $(s_1, \boldsymbol{a}_1, \{r_{i1}\}_i, \dots, s_{h+1}, \boldsymbol{a}_{h+1}, \{r_{i,h+1}\}_i \dots)$
       - If $(s_h, h) \in W$: update bandit instances at (s_h, h) with reward $r_{ih} + V_{i,h+1}(s_{h+1})$
       - If $(s_h, h) \notin W$: update bandit instances at (s_h, h) with reward $H + 1 - h$ (**Rmax reward**)
     - C. Define $V_{ih}(s)$ for all s as average of rewards given to bandit instance at s
   - Define $\tilde{\pi}^q$ as acting at each step h per the empirical average of the bandit instances in above procedure
   - If $\tilde{\pi}^q$ mostly only visits states in W: **output $\tilde{\pi}^q$, terminate** ☺
   - Else: for some "newly visited" state (h, s), set $\pi^{hs} \leftarrow \tilde{\pi}^q$, add (h, s) to W, continue with next stage

"backwards induction" idea from known model setting