Hardness of Independent Learning and Sparse Equilibrium Computation in Markov Games

Dylan J. Foster [1] Noah Golowich [2] Sham M. Kakade [3]

Abstract

We consider the problem of decentralized multi-agent reinforcement learning in Markov games. A key question is whether there are algorithms that, when run independently by all agents, lead to no-regret for each player, analogous to celebrated results for no-regret learning in normal-form games. While recent work has shown that such algorithms exist for restricted settings (e.g., when regret is defined with respect to deviations to Markov policies), the question of whether independent no-regret learning can be achieved in the standard Markov game framework was open. We provide a decisive negative resolution to this problem, both from a computational and statistical perspective. We show that:

1. Under the assumption that PPAD-hard problems cannot be solved in polynomial time, there is no polynomial-time algorithm that attains no-regret in general-sum Markov games when executed independently by all players, even when the game is known to the algorithm designer and the number of players is a small constant.
2. When the game is unknown, no algorithm, efficient or otherwise, can achieve no-regret without observing exponentially many episodes in the number of players.

These results are proven via lower bounds for a simpler problem we refer to as SPARSECCE, in which the goal is to compute a coarse correlated equilibrium that is "sparse" in the sense that it can be represented as a mixture of a small number of product policies.

[1]Microsoft Research [2]Massachusetts Institute of Technology [3]Harvard University. Correspondence to: Dylan J. Foster<dylanfoster@microsoft.com>, Noah Golowich<nzg@mit.edu>, Sham M. Kakade<sham@seas.harvard.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

1. Introduction

The framework of multi-agent reinforcement learning (MARL), which describes settings in which multiple agents interact in a dynamic environment, has played a key role in recent breakthroughs in artificial intelligence, including the development of agents that approach or surpass human performance in games such as Go (Silver et al., 2016), Poker (Brown & Sandholm, 2018), Stratego (Perolat et al., 2022), and Diplomacy (Kramar´et al., 2022; Bakhtin et al., 2022). MARL also shows promise for real-world multi-agent systems, including autonomous driving (Shalev-Shwartz et al., 2016), and cybersecurity (Malialis & Kudenko, 2015), and economic policy (Zheng et al., 2022). These applications, where reliability is critical, necessitate the development of algorithms that are practical and efficient, yet provide strong formal guarantees and robustness.

Multi-agent reinforcement learning is typically studied using the framework of Markov games (also known as stochastic games) (Shapley, 1953). In a Markov game, agents interact over a finite number of steps: at each step, each agent observes the state of the environment, takes an action, and observes a reward which depends on the current state as well as the other agents' actions. Then the environment transitions to a new state as a function of the current state and the actions taken. An episode consists of a finite number of such steps, and agents interact over the course of multiple episodes, progressively learning new information about their environment. Markov games generalize the well-known model of Markov Decision Processes (MDPs) (Puterman, 1994), which describe the special case in which there is a single agent acting in a dynamic environment, and we wish to find a policy that maximizes its reward. By contrast, for Markov games, we typically aim to find a distribution over agents' policies which constitutes some type of equilibrium.

1.1. Decentralized learning

In this paper, we focus on the problem of decentralized (or, independent) learning in Markov games. In decentralized MARL, each agent in the Markov game behaves independently, optimizing their policy myopically while treating the effects of the other agents as exogenous. Agents observe local information (in particular, their own actions and rewards), but do not observe the actions of the other agents directly. Decentralized learning enjoys a number of desirable properties, including

scalability, versatility, and practicality. The central question we consider is whether there exist decentralized learning algorithms which, when employed by all agents in a Markov game, lead them to play near-equilibrium strategies over time.

Decentralized equilibrium computation in MARL is not well understood theoretically, and algorithms with provable guarantees are scarce. To motivate the challenges and most salient issues, it will be helpful to contrast with the simpler problem of decentralized learning in normal-form games, which may be interpreted as Markov games with a single state. Much of the modern work on decentralized learning in normal-form games centers on no-regret learning, where agents select actions independently using online learning algorithms (Cesa-Bianchi & Lugosi, 2006) designed to minimize their regret (that is, the gap between realized payoffs and the payoff of the best fixed action in hindsight). In particular, a foundational result is that if each agent employs a no-regret learning strategy, then the average of the agents' joint action distributions approaches a coarse correlated equilibrium (CCE) for the normal-form game (Cesa-Bianchi & Lugosi, 2006; Hannan, 1957; Blackwell, 1956). CCE is a natural relaxation of the foundational concept of Nash equilibrium, which has the downside of being intractable to compute. On the other hand, there are many efficient algorithms that can achieve vanishing regret in a normal-form game, even when opponents select their actions in an arbitrary, potentially adaptive fashion, and thus converge to a CCE (Vovk, 1990; Littlestone & Warmuth, 1994; Cesa-Bianchi et al., 1997; Hart & Mas-Colell, 2000; Syrgkanis et al., 2015).

This simple connection between no-regret learning and decentralized convergence to equilibria has been influential in game theory, leading to numerous lines of research including fast rates of convergence to equilibria (Syrgkanis et al., 2015; Chen & Peng, 2020; Daskalakis et al., 2021; Anagnostides et al., 2022), price of anarchy bounds for smooth games (Roughgarden, 2015), and lower bounds on query and communication complexity for equilibrium computation (Fearnley et al., 2013; Rubinstein, 2016; Babichenko & Rubinstein, 2017). Empirically, no-regret algorithms such as regret matching (Hart & Mas-Colell, 2000) and Hedge (Vovk, 1990; Littlestone & Warmuth, 1994; Cesa-Bianchi et al., 1997) have been used to compute equilibria that can achieve state-of-the-art performance in application domains such as Poker (Brown & Sandholm, 2018) and Diplomacy (Bakhtin et al., 2022). Motivated by these successes, we ask whether an analogous theory can be developed for Markov games. In particular:

Are there efficient algorithms
for no-regret learning in Markov games?

Challenges for no-regret learning. In spite of active research effort and many promising pieces of progress (Jin et al., 2021; Song et al., 2022; Mao & Basar, 2021; Daskalakis et al., 2022; Erez et al., 2022), no-regret learning guarantees for Markov games have been elusive. A barrier faced by naive algorithms is that it is intractable to ensure no-regret against an arbitrary adversary, both computationally (Bai et al., 2020; Abbasi Yadkori et al., 2013) and statistically (Liu et al., 2022; Kwon et al., 2021; Foster et al., 2022). Fortunately, many of the implications of no-regret learning (in particular, convergence to equilibria) do not require the algorithm to have sublinear regret against an arbitrary adversary, but rather only against other agents who are running the same algorithm independently. This observation has been influential in normal-form games, where the line of work on fast rates of convergence to equilibrium (Syrgkanis et al., 2015; Chen & Peng, 2020; Daskalakis et al., 2021; Anagnostides et al., 2022) holds only in this more restrictive setting. This motivates the following relaxation to our central question.

Problem 1.1. Is there an efficient algorithm that, when adopted by all agents in a Markov game and run independently, leads to sublinear regret for each individual agent?

Attempts to address Problem 1.1. Two recent lines of research have made progress toward addressing Problem 1.1 and related questions. In one direction, several recent papers have provided algorithms, including V-learning (Jin et al., 2021; Song et al., 2022; Mao & Basar, 2021) and SPoCMAR (Daskalakis et al., 2022), that do not achieve no-regret, but can nevertheless compute and then sample from a coarse correlated equilibrium in a Markov game in a (mostly) decentralized fashion, with the caveat that they require a shared source of random bits as a mechanism to coordinate. Notably, V-learning depends only mildly on the shared randomness: agents first play policies in a fully independent fashion (i.e., without shared randomness) according to a simple learning algorithm for T episodes, and use shared random bits only once learning finishes as part of a post-processing procedure to extract a CCE policy. A question left open by these works, is whether the sequence of policies played by the V-learning algorithm in the initial independent phase can itself guarantee each agent sublinear regret.

Most closely related to our work, Erez et al. (2022) recently showed that Problem 1.1 can be solved positively for a restricted setting in which regret for each agent is defined as the maximum gain in value they can achieve by deviating to a fixed Markov policy. Markov policies are those whose choice of action depends only on the current state as opposed to the entire history of interaction. This notion of deviation is restrictive because in general, even when the opponent plays a sequence of Markov policies, the best response will be non-Markov. In challenging settings that abound in practice, it is standard to consider non-Markov policies (Leibo et al., 2021; Agapiou et al., 2022), since they often achieve higher value than Markov policies; we provide a simple example in Proposition B.1. Thus, while a regret guarantee with respect to the class of Markov policies (as in (Erez et al., 2022)) is certainly interesting, it may be too weak in general, and it is of great interest to understand whether Problem 1.1 can be answered positively in the general setting.[1]

[1]We remark that the V-learning and SPoCMAR algorithms

We refer the reader to Appendix B.2 for further discussion.

1.2. Our contributions

We resolve Problem 1.1 in the negative, from both a computational and statistical perspective.

Computational hardness. We provide two computational lower bounds (Theorems 1.2 and 1.3) which show that under standard complexity-theoretic assumptions, there is no efficient algorithm that runs for a polynomial number of episodes and guarantees each agent non-trivial ("sublinear") regret when used in tandem by all agents. Both results hold even if the Markov game is explicitly known to the algorithm designer; Theorem 1.3 is stronger and more general, but applies only to 3-player games, while Theorem 1.2 applies to 2-player games, but only for agents restricted to playing Markovian policies.

To state our first result, Theorem 1.2, we define a product Markov policy to be a joint policy in which players choose their actions independently according to Markov policies (see Sections 2 and 3 for formal definitions). Note that if all players use independent no-regret algorithms to choose Markov policies at each episode, then their joint play at each round is described by a product Markov policy, since any randomness in each player's policy must be generated independently.

Theorem 1.2 (Informal version of Corollary 3.3). If $\mathrm{PPAD} \subseteq \mathrm{P}$, then there is no polynomial-time algorithm that, given the description of a 2-player Markov game, outputs a sequence of joint product Markov policies which guarantees each agent sublinear regret.

Theorem 1.2 provides a decisive negative resolution to Problem 1.1 under the assumption that $\mathrm{PPAD} \not\subseteq \mathrm{P}$,[2] which is standard in the theory of computational complexity (Papadimitriou, 1994).[3] Beyond simply ruling out the existence of fully decentralized no-regret algorithms, it rules out existence of centralized algorithms that compute a sequence of product policies for which each agent has sublinear regret, even if such a sequence does not arise naturally as the result of agents independently following some learning algorithm. Salient implications include:

- Theorem 1.2 provides a separation between Markov games and normal-form games, since standard no-regret algorithms for normal-form games i) run in polynomial time and ii) produce sequences of joint product policies that guarantee each agent sublinear regret. Notably, no-regret learning for normal-form games is efficient whenever the number of agents is polynomial, whereas Theorem 1.2 rules out polynomial-time algorithms for as few as two agents.
- A question left open by the work of Jin et al. (2021); Song et al. (2022); Mao & Basar (2021) was whether the sequence of policies played by the V-learning algorithm during its independent learning phase can guarantee each agent sublinear regret. Since V-learning plays product Markov policies during the independent phase and is computationally efficient, Theorem 1.2 implies that these policies do not enjoy sublinear regret (assuming $\mathrm{PPAD} \not\subseteq \mathrm{P}$).

Our second result, Theorem 1.3, extends the guarantee of Theorem 1.2 to the more general setting in which agents can select arbitrary, potentially non-Markovian policies at each episode. This comes at the cost of only providing hardness for 3-player games as opposed to 2-player games, as well as relying on the slightly stronger complexity-theoretic assumption that $\mathrm{PPAD} \not\subseteq \mathrm{RP}$.[4]

Theorem 1.3 (Informal version of Corollary 4.4). If $\mathrm{PPAD} \not\subseteq \mathrm{RP}$, then there is no polynomial-time algorithm that, given the description of a 3-player Markov game, outputs a sequence of joint product general policies (i.e., potentially non-Markov) which guarantees each agent sublinear regret.

Statistical hardness. Theorems 1.2 and 1.3 rely on the widely-believed complexity theoretic assumption that PPAD-complete problems cannot be solved in (randomized) polynomial time. Such a restriction is inherent if we assume that the game is known to the algorithm designer. To avoid complexity-theoretic assumptions, we consider a setting in which the Markov game is unknown to the algorithm designer, and algorithms must learn about the game by executing policies ("querying") and observing the resulting sequences of states, actions, and rewards. Our final result, Theorem 1.4, shows unconditionally that, for m-player Markov games whose parameters are unknown, any algorithm computing a no-regret sequence as in Theorem 1.3 requires a number of queries that is exponential in m.

Theorem 1.4 (Informal version of Theorem 5.2). Given query access to a m-player Markov game, no algorithm that makes fewer than 2^{pmq} queries can output a sequence of joint product policies which guarantees each agent sublinear regret.

Similar to our computational lower bounds, Theorem 1.4 goes far beyond decentralized algorithms, and rules out even centralized algorithms that compute a no-regret sequence by ~~jointly controll~~ing all players. The result provides another

mentioned above do learn equilibria that are robust to deviations to non-Markov policies, though they do not address Problem 1.1 since they do not have sublinear regret.

[2] Technically, the class we are denoting by P, namely of total search problems that have a deterministic polynomial-time algorithm, is sometimes denoted by FP, as it is a search problem. We ignore this distinction.

[3] PPAD is the most well-studied complexity class in algorithmic game theory, and is widely believed to not admit polynomial time algorithms. Notably, the problem of computing a Nash equilibrium for normal-form games with two or more players is PPAD-complete (Daskalakis et al., 2009; Chen et al., 2006; Rubinstein, 2018).

[4] We use RP to denote the class of total search problems for which there exists a polynomial-time randomized algorithm which outputs a solution with probability at least $2/3$, and otherwise outputs "fail".

separation between Markov games and normal-form games, since standard no-regret algorithms for normal-form games can achieve sublinear regret using polypmq queries for any m. The 2^{pmq} scaling in the lower bound, which does not rule out query-efficient algorithms when m is constant, is to be expected for an unconditional result: If the game has only polynomially many parameters (which is the case for constant m), one can estimate all of the parameters using standard techniques (Jin et al., 2020), then directly find a no-regret sequence.

Proof techniques: the SPARSECCE problem. Our proofs proceed via establishing lower bounds for a computational problem we refer to as SPARSECCE. In the SPARSECCE problem, the aim is to compute a CCE that can be represented as the mixture of a small number of product policies. See Sections 3 and 4 for detailed proof overview.

Organization. Section 2 presents preliminaries, Sections 3 and 4 provide our computational lower bounds, and Section 5 presents our unconditional lower bounds for multi-player games.

Notation. For $n \in \mathbb{N}$, we write $[n] := \{1, 2, \ldots, n\}$. For a finite set T, $\Delta(T)$ denotes the space of distributions on T. For an element $t \in T$, $\mathbb{I}_t \in \Delta(T)$ denotes the delta distribution that places probability mass 1 on t. We adopt standard big-oh notation, and write $f \in \mathrm{poly}(g)$ to denote that $f \in O(g^r + \max\{1, \mathrm{polylog}(g)\}^r)$, with r (p,q) and (p,q) defined analogously.

2. Preliminaries

This section contains preliminaries necessary to present our main results. We first introduce the Markov game framework (Section 2.1), then provide a brief review of normal-form games (Section 2.3), and finally introduce the concepts of coarse correlated equilibria and regret minimization (Section 2.4).

2.1. Markov games

We consider general-sum Markov games in a finite-horizon, episodic framework. For $m \in \mathbb{N}$, an m-player Markov game G consists of a tuple $G = (S, H, (A_i)_{i \in [m]}, P, (R_i)_{i \in [m]}, \rho)$, where:

- S denotes a finite state space and $H \in \mathbb{N}$ denotes a finite time horizon. We write $S := |S|$.
- For $i \in [m]$, A_i denotes a finite action space for agent i. We let $A := \bigotimes_{i=1}^{m} A_i$ denote the joint action space and $A_{-i} := \bigotimes_{i' \neq i} A_{i'}$. We denote joint actions in bold, e.g., $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbf{A}$. We write $A_i := |A_i|$ and $A := |A|$.
- $P = (P_1, \ldots, P_H)$ is the transition kernel, with each $P_h : S \times A \to \Delta(S)$ denoting the kernel for step $h \in [H]$. In particular, $P_h(s' \mid s, a)$ is the probability of transitioning to s' from the state s at step h when agents play $\mathbf{a}$.
- For $i \in [m]$ and $h \in [H]$, $R_{i,h} : S \times \mathbf{A} \to [0, 1]$ is the reward function for agent i:[5] the reward agent i receives in state s at step h if agents play $\mathbf{a}$ is $R_{i,h}(s, \mathbf{a})$.[6]
- $\rho(S)$ denotes the initial state distribution.

An episode in the Markov game proceeds as follows: the initial state s_1 is drawn from the initial state distribution ρ. Then, for each $h = 1, \ldots, H$, given state s_h, each agent i plays action $a_{i,h} \in A_i$, and given the joint action profile $\mathbf{a}_h = (a_{1,h}, \ldots, a_{m,h})$, each agent i receives reward of $r_{i,h} = R_{i,h}(s_h, \mathbf{a}_h)$ and the state of the system transitions to $s_{h+1} \sim P_h(\cdot \mid s_h, \mathbf{a}_h)$. We denote the tuple of agents' rewards at each step h by $\mathbf{r}_h = (r_{1,h}, \ldots, r_{m,h})$, and refer to the resulting sequence $\tau_H = (s_1, \mathbf{a}_1, \mathbf{r}_1, \ldots, s_H, \mathbf{a}_H, \mathbf{r}_H)$ as a trajectory. For $h \in [H]$, we define the prefix of the trajectory via $\tau_h = (s_1, \mathbf{a}_1, \mathbf{r}_1, \ldots, s_h, \mathbf{a}_h, \mathbf{r}_h)$.

We use the following notation: for some quantity x (e.g., action, reward, etc.) indexed by agents, i.e., $x = (x_1, \ldots, x_m)$, and an agent $i \in [m]$, we write $x_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m)$ to denote the tuple consisting of all $x_{i'}$ for $i' \neq i$.

2.2. Policies and value functions

We now introduce the notion of policies and value functions for Markov games. Policies are mappings from states (or sequences of states) to actions for the agents. We consider several different types of policies, which play a crucial role in distinguishing the types of equilibria that are tractable and those that are intractable to compute efficiently.

Markov policies. A randomized Markov policy for agent i is a sequence $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,H})$, where $\pi_{i,h} : S \to \Delta(A_i)$. We denote the space of randomized Markov policies for agent i by Π_i^{markov}. We write $\Pi^{\mathrm{markov}} := \Pi_1^{\mathrm{markov}} \times \cdots \times \Pi_m^{\mathrm{markov}}$ to denote the space of product Markov policies, which are joint policies in which each agent i independently follows a policy in Π_i^{markov}. In particular, a policy $\pi \in \Pi^{\mathrm{markov}}$ is specified by a collection $\pi = (\pi_1, \ldots, \pi_H)$, where $\pi_h : S \to \Delta(A_1) \times \cdots \times \Delta(A_m)$. We additionally define $\Pi_{-i}^{\mathrm{markov}} := \bigotimes_{i' \neq i} \Pi_{i'}^{\mathrm{markov}}$, and for a policy $\pi \in \Pi^{\mathrm{markov}}$, write π_{-i} to denote the collection of mappings $\pi_{-i} = (\pi_{-i,1}, \ldots, \pi_{-i,H})$, where $\pi_{-i,h} : S \to \prod_{i' \neq i} \Delta(A_{i'})$. π_{-i} denotes the tuple of all but player i's policies.

When the Markov game G is clear from context, for a policy $\pi \in \Pi^{\mathrm{markov}}$ we let $\Pr_\pi$ denote the law of the trajectory when players select actions via $\mathbf{a}_h \sim \pi_h(s_h)$, and let $\mathbb{E}_\pi$ denote the corresponding expectation.

General (non-Markov) policies. In addition to Markov policies, we will consider general history-dependent (or, non-

[5] We assume that rewards lie in $[0, 1]$ for notational convenience, as this ensures that the cumulative reward for each episode lies in $[0, 1]$. This assumption is not important to our results.

[6] We restrict our attention to Markov games in which the rewards at each step are a deterministic function of the state and action profile. Since our goal is to prove lower bounds, this is without loss.

Markov) policies, which select actions based on the entire sequence of states and actions observed up the current step. To streamline notation, for $i \in [m]$, let $\tau_{i,h} = (s_1, a_{i,1}, r_{i,1}, \ldots, s_h, a_{i,h}, r_{i,h})$ denote the history of agent i's states, actions, and reward up to step h. Let $\mathcal{H}_{i,h} = (\mathcal{S} \times \mathcal{A}_i \times [0,1])^h$ denote the space of all possible histories of agent i up to step h. For $i \in [m]$, a randomized general (i.e., non-Markov) policy of agent i is a collection of mappings $\pi_i = (\pi_{i,1}, \ldots, \pi_{i,H})$ where $\pi_{i,h} : \mathcal{H}_{i,h-1} \times \mathcal{S} \to \Delta(\mathcal{A}_i)$ is a mapping that takes the history observed by agent i up to step $h-1$ and the current state and outputs a distribution over actions for agent i.

We denote by $\Pi_i^{\mathsf{gen,rnd}}$ the space of randomized general policies of agent i, and further write $\Pi^{\mathsf{gen,rnd}} := \Pi_1^{\mathsf{gen,rnd}} \times \cdots \times \Pi_m^{\mathsf{gen,rnd}}$ to denote the space of product general policies; note that $\Pi^{\mathsf{markov}} \subseteq \Pi^{\mathsf{gen,rnd}}$ and $\Pi_i^{\mathsf{markov}} \subseteq \Pi_i^{\mathsf{gen,rnd}}$. In particular, a policy $\pi \in \Pi^{\mathsf{gen,rnd}}$ is specicfied by a collection $(\pi_{i,h})_{i \in [m], h \in [H]}$, where $\pi_{i,h} : \mathcal{H}_{i,h-1} \times \mathcal{S} \to \Delta(\mathcal{A}_i)$. When agents play according to a general policy $\pi \in \Pi^{\mathsf{gen,rnd}}$, at each step h, each agent, given the current state s_h and their history $\tau_{i,h-1} \in \mathcal{H}_{i,h-1}$, chooses to play an action $a_{i,h} \sim \pi_{i,h}(\tau_{i,h-1}, s_h)$, independently from all other agents. For a policy $\pi \in \Pi^{\mathsf{gen,rnd}}$, we let $\Pr_\pi$ and $\mathbb{E}_\pi$ denote the law and expectation operator for the trajectory when players select actions via $a_h \sim \pi_h(s_h)$, and write π_{-i} to denote the collection of policies of all agents but i, i.e., $\pi_{-i} = (\pi_{j,h})_{h \in [H], j \in [m], j \neq i}$.

We will also consider distributions over product randomized general policies, namely elements of $\Delta(\Pi^{\mathsf{gen,rnd}})$.[7] We will refer to elements of $\Delta(\Pi^{\mathsf{gen,rnd}})$ as distributional policies. To play a distributional policy $\Pi \in \Delta(\Pi^{\mathsf{gen,rnd}})$, agents draw a randomized policy $\pi \sim \Pi$ (so that $\pi \in \Pi^{\mathsf{gen,rnd}}$) and then play π.

Value functions. For a general policy $\pi \in \Pi^{\mathsf{gen,rnd}}$, we define the value function for agent $i \in [m]$ as $V_i^\pi := \mathbb{E}_\pi \sum_{h=1}^H R_{i,h}(s_h, a_h) \mid s_1$; this represents the expected reward that agent i receives when each agent chooses their actions via $a_{i,h} \sim \pi_h(\tau_{i,h-1}, s_h)$. For a distributional policy $\Pi \in \Delta(\Pi^{\mathsf{gen,rnd}})$, we extend this notation by defining $V_i^\Pi := \mathbb{E}_{\pi \sim \Pi} V_i^\pi$.

2.3. Normal-form games

To motivate the solution concepts we consider for Markov games, let us first revisit the notion of normal-form games, which may be interpreted as Markov games with a single state. For $m, n \in \mathbb{N}$, an m-player n-action normal-form game G is specified by a tuple of m reward tensors $M_1, \ldots, M_m \in [0,1]^{n \times \cdots \times n}$, where each tensor is of order m (i.e., has n^m entries). We will write $G = (M_1, \ldots, M_m)$. We assume for simplicity that each player has the same number n of actions, and identify each player's action space with $[n]$. Then an an action profile is specified by $a \in [n]^m$; if each player acts according to a, then the reward for player $i \in [m]$ is given by $M_i(a) \in [0,1]$. Our hardness results will use the standard notion of Nash equilibrium in normal-form games. We define the m-player (n, ε)-NASH problem to be the problem of computing an ε-approximate Nash equilibrium of a given m-player n-action normal-form game. (See Definition C.2 for a formal definition of ε-Nash equilibrium.) A celebrated result is that Nash equilibria are PPAD-hard to approximate, i.e., the 2-player (n, n^{-c})-NASH problem is PPAD-hard for any constant $c > 0$ (Daskalakis et al., 2009; Chen et al., 2006). We refer the reader to Section C.2 for further background on these concepts.

2.4. Markov games: Equilibria and no-regret

We now turn our focus back to Markov games, and introduce the main solution concepts we consider, as well as the notion of no-regret. Since computing Nash equilibria is intractable even for normal-form games, much of the work on efficient equilibrium computation has focused on alternative notions of equilibrium, notably coarse correlated equilibria.

For a distributional policy $\Pi \in \Delta(\Pi^{\mathsf{gen,rnd}})$ and a randomized policy $\pi_i \in \Pi_i^{\mathsf{gen,rnd}}$ of player i, we let $(\pi_i \times \Pi_{-i}) \in \Delta(\Pi^{\mathsf{gen,rnd}})$ denote the distributional policy which is given by the distribution of (π_i, π_{-i}) for $\pi_{-i} \sim \Pi_{-i}$ (and Π_{-i} denotes the marginal of Π on all players but i). For $\pi \in \Pi^{\mathsf{gen,rnd}}$, we write π_{-i} to denote the policy given by $(\pi_j)_{j \neq i}$. Let us fix a Markov game G, which in particular determines the players' value functions V_i^π.

Definition 2.1 (Coarse correlated equilibrium). For $\varepsilon \geq 0$, a distributional policy $\Pi \in \Delta(\Pi^{\mathsf{gen,rnd}})$ is defined to be an ε-coarse correlated equilibrium (CCE) if for each $i \in [m]$, it holds that $\max_{\pi_i \in \Pi^{\mathsf{gen,rnd}}} V_i^{(\pi_i \times \Pi_{-i})} \leq V_i^\Pi + \varepsilon$.

Coarse correlated equilibria can be computed efficiently for both normal-form games and Markov games, and are fundamentally connected to the notion of no-regret and independent learning, which we now introduce.

Regret. For a policy $\pi \in \Pi^{\mathsf{gen,rnd}}$, we denote the distributional policy which puts all its mass on π by $\mathbb{I}_\pi \in \Delta(\Pi^{\mathsf{gen,rnd}})$. Thus $\frac{1}{T} \sum_{t=1}^T \mathbb{I}_{\pi^t} \in \Delta(\Pi^{\mathsf{gen,rnd}})$ denotes the distributional policy which randomizes uniformly over the π^t. We define regret as follows.

Definition 2.2 (Regret). Consider a sequence of policies $\pi^1, \ldots, \pi^T \in \Pi^{\mathsf{gen,rnd}}$. For $i \in [m]$, the regret of agent i with respect to this sequence is defined as:

$$\mathrm{Reg}_{i,T}(\pi^1, \ldots, \pi^T) := \max_{\pi_i \in \Pi_i^{\mathsf{gen,rnd}}} \sum_{t=1}^T \left(V_i^{(\pi_i \times \pi_{-i}^t)} - V_i^{\pi^t} \right). \quad (1)$$

It is immediate from the above definitions that a sequence of policies $\pi^1, \ldots, \pi^T \in \Pi^{\mathsf{gen,rnd}}$ satisfies $\mathrm{Reg}_{i,T}(\pi^1, \ldots, \pi^T) / T \leq \varepsilon$

[7] When $\mathcal{T}$ is not a finite set, we take $\Delta(\mathcal{T})$ to be the set of Radon probability measures over $\mathcal{T}$ equipped with the Borel σ-algebra.

T if and only if the distributional policy $\frac{1}{T}\sum_{t=1}^{T} I_{\pi^{t,q}}$ is an ε-CCE (stated formally in Fact C.1 in the appendix).

No-regret learning. A standard approach to decentralized equilibrium computation, which exploits Fact C.1, is to select $\pi^{1,q}, \ldots, \pi^{T,q} \in \Pi^{gen,rnd}$ using independent no-regret learning algorithms. A no-regret learning algorithm for player i selects $\pi_i^{t,q} \in \Pi_i^{gen,rnd}$ based on the realized trajectories $\pi_i^{1,q}, \ldots, \pi_i^{t-1,q} \in \Pi_{i,h}, H$ that player i observes over the course of play,[8] but with no knowledge of $\pi_{-i}^{t,q}$, so as to ensure that no-regret is achieved: $\mathrm{Reg}_{i,T}(\pi^{1,q}, \ldots, \pi^{T,q})/T$. If each player i uses their own, independent no-regret learning algorithm, this approach yields product policies $\pi^{t,q} = \pi_1^{t,q} \times \cdots \times \pi_m^{t,q}$, and the uniform average of the $\pi^{t,q}$ yields a CCE as long as all of the players can keep their regret small.[9]

For the special case of normal-form games, there are several efficient algorithms, which—when run independently—ensure that each player's regret after T episodes is bounded above by $O(\sqrt{T})$ (that is, $O(1/\sqrt{T})$), even when the other players' actions are chosen adversarially.

3. Lower bound for Markovian algorithms

In this section we prove Theorem 1.2 (restated formally below as Theorem 3.2), establishing that in two-player Markov games, there is no computationally efficient algorithm that computes a sequence $\pi^{1,q}, \ldots, \pi^{T,q}$ of product Markov policies so that each player has small regret under this sequence. This section serves as a warm-up for our results in Section 4, which remove the assumption that $\pi^{1,q}, \ldots, \pi^{T,q}$ are Markovian.

3.1. SparseMarkovCCE and computational model

As discussed in the introduction, our lower bounds for no-regret learning are a consequence of lower bounds for the SparseCCE problem. In what follows, we formalize this problem (specifically, the Markovian variant, which we refer to as SparseMarkovCCE), as well as our computational model.

Description length for Markov games (constant m). Given a Markov game G, we let $\langle G \rangle$ denote the maximum number of bits needed to describe any of the rewards $R_{i,h}(s,a)$ or transition probabilities $P_h(s' \mid s,a)$ in binary.[10] We define $|G| := \max\{S, \max_{i \in [m]} A_i, H, \langle G \rangle\}$. The interpretation of $|G|$ depends on the number of players m: If m is a constant (as will be the case in the current section and Section 4), then $|G|$ should be interpreted as the description length of the game G, up to polynomial factors. In particular, for constant m, the game G can be described using $|G|^{O(1)}$ bits. In Section 5, we discuss the interpretation of $|G|$ when m is large.

The SparseMarkovCCE problem. From Fact C.1, we know that the problem of computing a sequence $\pi^{1,q}, \ldots, \pi^{T,q}$ of joint product Markov policies for which each player has at most T regret is equivalent to computing a sequence $\pi^{1,q}, \ldots, \pi^{T,q}$ for which the uniform mixture forms an ε-approximate CCE. We define (T, ε)-SparseMarkovCCE as the computational problem of computing such a CCE directly.

Definition 3.1 (SparseMarkovCCE problem). For an m-player Markov game G and parameters $T \in \mathbb{N}$ and $\varepsilon \geq 0$ (which may depend on the size of the game G), (T, ε)-SparseMarkovCCE is the problem of finding a sequence $\pi^{1,q}, \ldots, \pi^{T,q}$, with each $\pi^{t,q} \in \Pi^{markov}$, such that the distributional policy $\frac{1}{T}\sum_{t=1}^{T} I_{\pi^{t,q}} \in \Pi^{gen,rnd}$ is an ε-CCE of G (or equivalently, such that for all $i \in [m]$, $\mathrm{Reg}_{i,T}(\pi^{1,q}, \ldots, \pi^{T,q})/T \leq \varepsilon$).

Decentralized learning algorithms naturally lead to solutions to the SparseMarkovCCE problem. In particular, consider any decentralized protocol which runs for T episodes, where at each timestep $t \in [T]$, each player $i \in [m]$ chooses a Markov policy $\pi_i^{t,q} \in \Pi_i^{markov}$ to play, without knowledge of the other players' policies $\pi_{-i}^{t,q}$ (but possibly using the history); any strategy in which players independently run online learning algorithms falls under this protocol. If each player experiences overall regret at most T, then the sequence $\pi^{1,q}, \ldots, \pi^{T,q}$ is a solution to the (T, ε)-SparseMarkovCCE problem. However, one might expect the (T, ε)-SparseMarkovCCE problem to be much easier than decentralized learning, since it allows for algorithms that produce $\pi^{1,q}, \ldots, \pi^{T,q}$ satisfying the constraints of Definition 3.1 in a centralized manner. The main result of this section, Theorem 3.2, rules out the existence of any efficient algorithms, including centralized ones, that solve the SparseMarkovCCE problem.

Before moving on, let us give a sense for what sort of scaling one should expect for the parameters T and ε in the (T, ε)-SparseMarkovCCE problem. First, we note that there always exists a solution to the $(1, 0)$-SparseMarkovCCE problem in a Markov game, which is given by a (Markov) Nash equilibrium of the game; of course, Nash equilibria are intractable to compute in general.[11] For the special case of normal-form games (where there is only a single state, and $H = 1$), no-regret learning (e.g., Hedge) yields a computationally efficient solution to the $(T, O(1/\sqrt{T}))$-SparseMarkovCCE problem, where the $O(\cdot)$ hides a $\max_i \log |A_i|$ factor. Refined convergence guarantees of Daskalakis et al. (2021); Anagnostides et al. (2022) improve upon this result, and yield an efficient solution

[8] An alternative model allows for player i to have knowledge of the previous joint policies $\pi^{1,q}, \ldots, \pi^{t-1,q}$, when selecting $\pi_i^{t,q}$.

[9] In Appendix B, we discuss the implications of relaxing the stipulation that $\pi_i^{t,q}$ be product policies (for example, by allowing the use of shared randomness, as in V-learning). In short, allowing $\pi_i^{t,q}$ to be non-product essentially trivializes the problem.

[10] We emphasize that $\langle G \rangle$ is defined as the maximum number of bits required by any particular (s,a) pair, not the total number of bits required for all (s,a) pairs.

[11] Such a Nash equilibrium can be seen to exist by using backwards induction to specify the player's joint distribution of play at each state at steps $H, H-1, \ldots, 1$.

to the $(T, \frac{1}{\mathrm{poly}(T)})$-SparseMarkovCCE problem.

3.2. Main result

Theorem 3.2. There is a constant $C_0 > 1$ so that the following holds. Let $n \in \mathbb{N}$ be given, and let $T \in \mathbb{N}$ and $\varepsilon > 0$ satisfy $T \leq \exp(n^{1/2}/2^5)$. Suppose there is an algorithm that, given the description of any 2-player Markov game G with $|G| = n$, solves the (T, ε)-SparseMarkovCCE problem in time U, for some $U \in \mathbb{N}$. Then, for each $n \in \mathbb{N}$, the 2-player $(n^{1/2}, 4)$-Nash problem (Definition C.2) can be solved in time $n \cdot (T + U) \cdot n^{C_0}$.

We emphasize that the range $T \leq \exp(n^{O(1)})$ ruled out by Theorem 3.2 is the most natural parameter regime, since the runtime of any decentralized algorithm which runs for T episodes and produces a solution to the SparseMarkovCCE problem is at least linear in T. Using that 2-player (n, ε)-Nash is PPAD-complete for n^c (for any $c > 0$) (Daskalakis et al., 2009; Chen et al., 2006; Rubinstein, 2018), we obtain the following corollary.

Corollary 3.3 (SparseMarkovCCE is PPAD-complete). For any constant $C > 4$, if there is an algorithm which, given the description of a 2-player Markov game G, solves the $(|G|^C, |G|^{-C})$-SparseMarkovCCE problem in time $\mathrm{poly}(|G|)$, then PPAD $= \mathrm{P}$.

The condition $C > 4$ in Corollary 3.3 is set to ensure that $|G|^C \leq \exp(|G|^{2/C}/2^6)$ for sufficiently large $|G|$, so as to satisfy the condition of Theorem 3.2. Corollary 3.3 rules out the existence of a polynomial-time algorithm that solves the SparseMarkovCCE problem with accuracy polynomially small and T polynomially large in $|G|$.

Proof overview. The proof of Theorem 3.2 is based on a reduction, which shows that any algorithm that efficiently solves the (T, ε)-SparseMarkovCCE problem, for T not too large, can be used to efficiently compute an approximate Nash equilibrium of any given normal-form game. In particular, fix $n_0 \in \mathbb{N}$, and let a 2-player normal form game G with n_0 actions be given.

We construct a Markov game $G = G(G)$ with horizon $H = n_0$ and action sets identical to those of the game G, i.e., $A_1 = A_2 = [n_0]$. The state space of G consists n_0^2 states, which are indexed by joint action profiles; the transitions are defined so that the value of the state at step h encodes the action profile taken by the agents at step $h-1$.[12] At each state of G, the reward functions are given by the payoff matrices of G, scaled down by a factor of $1/H$ (which ensures that the rewards received at each step belong to $[0, 1/H]$). In particular, the rewards and transitions out of a given state do not depend on the identity of the state, and so G can be thought of as a repeated game where G is played H times. The formal definition of G is given in Definition D.3.

Fix any algorithm for the SparseMarkovCCE problem, and recall that for each step h and state s for G, $\mu_h^{\mathrm{alg}}(s) \in \Delta(A_1 \times A_2)$ denotes the joint action distribution taken in s at step h for the sequence of $\pi^{(1)}, \ldots, \pi^{(T)}$ produced by the algorithm. The bulk of the proof of Theorem 3.2 consists of proving a key technical result, Lemma D.4, which states that if $\pi^{(1)}, \ldots, \pi^{(T)}$ indeed solves (T, ε)-SparseMarkovCCE, then there exists some tuple (h, s, t) such that $\mu_h^{\mathrm{alg}}(s)$ is an approximate Nash equilibrium for G. With this established, it follows that we can find a Nash equilibrium efficiently by simply trying all $H \cdot S \cdot T$ choices for (h, s, t).

To prove Lemma D.4, we reason as follows. Assume that $\bar{\pi} := \frac{1}{T}\sum_{t=1}^{T} \pi^{(t)} \in \Pi^{\mathrm{gen,rnd}}$ is an ε-CCE. If, by contradiction, none of the distributions $\mu_h^{\mathrm{alg}}(s) \in \Delta(\Pi^{\mathrm{H,S,T}})$ are approximate Nash equilibria for G, then it must be the case that for each t, one of the players has a profitable deviation in G with respect to the product strategy $\mu_h^{\mathrm{alg}}(s)$, at least for a constant fraction of the tuples (s, h). We will argue that if this were to be the case, it would imply that there exists a non-Markov deviation policy for at least one player i in Definition 2.1, meaning that $\bar{\pi}$ is not in fact an ε-CCE.

To sketch the idea, recall that to draw a trajectory from $\bar{\pi}$, we first draw a random index $t \in [T]$ uniformly at random, and then execute $\pi^{(t)}$ for an episode. We will show (roughly) that for each player i, it is possible to compute a non-Markov deviation policy $\dagger_i$ which, under the draw of a trajectory from $\bar{\pi}$, can "infer" the value of the index t within the first few steps of the episode. Then policy $\dagger_i$ then, at each state s and step h after the first few steps, play a best response to their opponent's portion of the strategy $\mu_h^{\mathrm{alg}}(s)$. If, for each possible value of t, none of the distributions $\mu_h^{\mathrm{alg}}(s)$ are approximate Nash equilibria of G, this means that at least one of the players i can significantly increase their value in G over that of $\bar{\pi}$ by playing $\dagger_i$, which contradicts the assumption that $\bar{\pi}$ is an ε-CCE.

It remains to explain how we can construct a non-Markov policy $\dagger_i$ which "infers" the value of t. Unfortunately, exactly inferring the value of t in the fashion described above is impossible: for instance, if there are $t_1 < t_2$ so that $\pi^{(t_1)} = \pi^{(t_2)}$, then clearly it is impossible to distinguish between the cases $t = t_1$ and $t = t_2$. Nevertheless, by using the fact that each player observes the full joint action profile played at each step h, we can construct a non-Markov policy which employs Vovk's aggregating algorithm for online density estimation (Vovk, 1990; Cesa-Bianchi & Lugosi, 2006) in order to compute a distribution which is close to $\mu_h^{\mathrm{alg}}(s)$ for most $h \in [H]$.[13] This guarantee is stated formally in an abstract setting in Proposition D.2, and is instantiated in the proof of Theorem 3.2 in ((5)). As we show in Section D.2, approximating $\mu_h^{\mathrm{alg}}(s)$ as we have described is sufficient to carry out the reasoning from the previous paragraph.

[12] For technical reasons, this only is the case for even values of h; we discuss further details in the full proof in Section D.2.

[13] Vovk's aggregating algorithm is essentially the exponential weights algorithm with the logarithmic loss. A detailed background for the algorithm is provided in Section D.1.

4. Lower bound for non-Markov algorithms

In this section, we prove Theorem 1.3 (restated formally below as Theorem 4.3), which strengthens Theorem 3.2 by allowing the sequence $\pi^{(1)},\ldots,\pi^{(T)}$ of product policies to be non-Markovian. This additional strength comes at the cost of our lower bound only applying to 3-player Markov games (as opposed to Theorem 3.2, which applied to 2-player games).

4.1. SparseCCE problem and computational model

To formalize the computational model for the SparseCCE problem, we must first describe how the non-Markov product policies $\pi^{(t)} = (\pi_1^{(t)},\ldots,\pi_m^{(t)})$ are represented. Recall that a non-Markov policy $\pi_i^{(t)} \in \Pi_i^{\mathrm{gen,rnd}}$ is, by definition, a mapping from agent i's history and current state to a distribution over their next action. Since there are exponentially many possible histories, it is information-theoretically impossible to express an arbitrary policy in $\Pi_i^{\mathrm{gen,rnd}}$ with polynomially many bits. As our focus is on computing a sequence of such policies $\pi^{(t)}$ in polynomial time, certainly a prerequisite is that $\pi^{(t)}$ can be expressed in polynomial space. Thus, we adopt the representational assumption, stated formally in Definition 4.1, that each of the policies $\pi_i^{(t)} \in \Pi_i^{\mathrm{gen,rnd}}$ is described by a bounded-size circuit that can compute the conditional distribution of each next action given the history. This assumption is satisfied by essentially all empirical and theoretical work concerning non-Markov policies (e.g., (Leibo et al., 2021; Agapiou et al., 2022; Jin et al., 2021; Song et al., 2022)).

Definition 4.1 (Computable policy). Given a m-player Markov game G and $N \in \mathbb{N}$, we say that a policy $\pi_i \in \Pi_i^{\mathrm{gen,rnd}}$ is N-computable if for each $h \in [H]$, there is a circuit of size N that,[14] on input $(s_{1:h-1}, s) \in (S \times A_i)^{h-1} \times S$, outputs the distribution $\pi_i(s_{1:h-1}, s) \in \Delta(A_i)$. A policy $(\pi_1,\ldots,\pi_m) \in \Pi^{\mathrm{gen,rnd}}$ is N-computable if each constituent policy π_i is.

Our lower bound applies to algorithms that produce sequences $\pi^{(1)},\ldots,\pi^{(T)}$ for which each $\pi^{(t)}$ is N-computable, where the value N is taken to be polynomial in the description length of the game G. For example, Markov policies whose probabilities can be expressed with b bits are $O(HSA_i(b+\log A_i))$-computable for each player i, since one can simply store each of the probabilities $\pi_{i,h}(s_h, a_{i,h})$, each of which takes b bits to represent.

The SparseCCE problem. SparseCCE is the problem of computing a sequence of non-Markov product policies $\pi^{(1)}, \ldots, \pi^{(T)}$ such that the uniform mixture forms an ε-approximate CCE. The problem generalizes SparseMarkovCCE (Definition 3.1) by relaxing the condition that the policies $\pi^{(t)}$ be Markov.

[14] For concreteness, we suppose that "circuit" means "boolean circuit" as in Definition 6.1 of (Arora & Barak, 2006), where probabilities are represented in binary. The precise model of computation we use does not matter, though, and we could equally assume that the policies π_i may be computed by Turing machines that terminate after N steps.

Definition 4.2 (SparseCCE Problem). For an m-player Markov game G and parameters $T, N \in \mathbb{N}$ and $\varepsilon > 0$ (which may depend on the size of the game G), (T, ε, N)-SparseCCE is the problem of finding a sequence $\pi^{(1)},\ldots,\pi^{(T)} \in \Pi^{\mathrm{gen,rnd}}$, with each $\pi^{(t)}$ being N-computable, such that the distributional policy $\frac{1}{T}\sum_{t=1}^{T} \mathbb{I}_{\pi^{(t)}} \in \Pi^{\mathrm{gen,rnd}}$ is an ε-CCE for G (equivalently, such that for all $i \in [m]$, $\mathrm{Reg}_{i,T}(\pi^{(1)},\ldots,\pi^{(T)})/T \le \varepsilon$).

4.2. Main result

Our main theorem for this section, Theorem 4.3, shows that for appropriate values of T, ε, and N, solving the (T, ε, N)-SparseCCE problem is at least as hard as computing Nash equilibria in normal-form games.

Theorem 4.3. Fix $n \in \mathbb{N}$, and let $T, N \in \mathbb{N}$, and $\varepsilon > 0$ satisfy $1 \le T \le \exp\left(\frac{n}{1-\varepsilon}\right)$. Suppose there exists an algorithm that, given the description of any 3-player Markov game G with $|G| = n$, solves the (T, ε, N)-SparseCCE problem in time U, for some $U \in \mathbb{N}$. Then, for any $\varepsilon > 0$, the 2-player $\mathrm{poly}(\{2, 50\}, \frac{1}{\varepsilon})$-Nash problem can be solved in randomized time $\mathrm{poly}(T, N, U, \log(1/\delta), 1/\varepsilon)$ with failure probability δ, where $C_0 > 0$ is an absolute constant.

By analogy to Corollary 3.3, we obtain the following immediate consequence.

Corollary 4.4 (SparseCCE is hard under PPAD $\subseteq$ RP). For any $C > 4$, if there is an algorithm which, given the description of a 3-player Markov game G, solves the $(|G|^C, |G|^{-C}, |G|^C)$-SparseCCE problem in time $\mathrm{poly}(|G|)$, then PPAD $\subseteq$ RP.

Proof overview for Theorem 4.3. The proof of Theorem 4.3 has a similar high-level structure to that of Theorem 3.2: given an m-player normal-form G, we define an $(m+1)$-player Markov game $G' = G'(G)$ which has $n_0 = \max\{m, A\}$ actions per player and horizon $H = n_0$. The key difference in the proof of Theorem 4.3 is the structure of the players' reward functions. To motivate this difference and the addition of an $(m+1)$-th player, we explain why the proof of Theorem 3.2 fails to extend: a sequence $\pi^{(1)},\ldots,\pi^{(T)}$ can hypothetically solve the SparseCCE problem by attempting to punish any one player's deviation policy, and thus avoid having to compute a Nash equilibrium of G. In particular, if player i plays according to the policy π_i' that we described in Section 3.2, then other players $j \neq i$ can use the non-Markov property of $\pi^{(t)}$ to adjust their choice of actions in later rounds to decrease player i's value.

This behavior is reminiscent of "tit-for-tat" strategies which are used to establish the folk theorem in the theory of repeated games (Maskin & Fudenberg, 1986). The folk theorem describes how Nash equilibria are more numerous in repeated games than in single-shot normal form games. As it turns out, the folk theorem does not yield to worst-case speedups in repeated games, when the number of players is at least 3. Indeed, Borgs et al. (2008) gave an "anti-folk theorem", showing that computing Nash equilibria in $(m+1)$-player

repeated games is PPAD-hard for m ¥ 2, via a reduction to m-player normal-form games. We adapt their reduction to our setting: roughly speaking, this approach adds an pm 1q-th player whose actions represent potential deviations for each of the m players. The structure of the rewards ensures that if $\frac{1}{T}\sum_{t=1}^{T} l_{pt_q}$ is an -CCE, then for some policy π_{m+1} of the pm 1q-th player, the first m players will play an approximate Nash of G with constant probability, under a trajectory drawn from the joint policy $\bar{\pi}_{pm+1q}$. Thus, in order to efficiently find a Nash (see Algorithm 2), we need to simulate the policy $\bar{\pi}_{pm+1q}$, which involves running Vovk's algorithm. This approach is in contrast to the proof of Theorem 3.2, which used Vovk's algorithm as an ingredient in the proof but not in the Nash computation algorithm.

Two-player games. One intruiging question we leave open is whether the SPARSECCE problem remains hard for two-player Markov games. Interestingly, as shown by Littman & Stone (2005), there is a polynomial time algorithm to find an exact Nash equilibrium for the special case of repeated two-player normal-form games. Though their result only applies in the infinite-horizon setting, it is possible to extend their results to the finite-horizon setting, which rules out naive approaches to extend the proof of Theorem 4.3 and Corollary 4.4 to two players.

5. Multi-player games: lower bounds

In this section we present Theorem 1.4 (restated formally below as Theorem 5.2), which gives a statistical lower bound for the SPARSECCE problem. The lower bound applies to any algorithm, regardless of computational cost, that accesses the underlying Markov game through a generative model.

Definition 5.1 (Generative model). For an m-player Markov game G pS;H;pA$_i$q$_{i\in[m]}$;P;pR$_i$q$_{i\in[m]}$;q, a generative model oracle is defined as follows: given a query described by a tuple ph;s;aq P rHsSA, the oracle returns the distribution P$_h$p|s;aq P pSq and the tuple of rewards pR$_{i;h}$ps;aqq$_{i\in[m]}$.

From the perspective of lower bounds, the assumption that the algorithm has access to a generative model is quite reasonable, as it encompasses most standard access models in RL, including the online access model, in which the algorithm repeatedly queries a policy and observes a trajectory drawn from it, as well as the local access generative model used in from (Yin et al., 2022; Weisz et al., 2021). We remark that it is slightly more standard to assume that queries to the generative model only return a sample from the distribution P$_h$p|s;aq as opposed to the distribution itself (Kakade, 2003; Kearns et al., 1999), but since our goal is to prove lower bounds, the notion in Definition 5.1 only makes our results stronger.

To state our main result, we recall the definition |G| maxtS; max$_{i\in[m]}$ A$_i$; H; pGqu. In the present section, we consider the setting where the number of players m is large. Here, |G| does not necessarily correspond to the description length for G, and should be interpreted, roughly speaking, as a measure of the description complexity of G |G| with respect to decentralized learning algorithms. In particular, from the perspective of an individual agent implementing a decentralized learning algorithm, their sample complexity should depend only on the size of their individual action set (as well as the global parameters S;H;pGq), as opposed to the size of the joint action set, which grows exponentially in m; the former is captured by jGj, while the latter is not. Indeed, a key advantage shared by much prior work on decentralized RL (Jin et al., 2021; Song et al., 2022; Mao & Basar, 2021; Daskalakis et al., 2022) is their avoidance of the curse of multi-agents, which describes the situation where an algorithm has sample and computational costs that scale exponentially in m.

Our main result for this section, Theorem 5.2, states that for m-player Markov games, exponentially many generative model queries (in m) are necessary to produce a solution to the pT;;Nq-SPARSECCE problem, unless T is exponential in m.

Theorem 5.2. Let m ¥ 2 be given. There are constants c; ¡ 0 so that the following holds. Suppose there is an algorithm B which, given access to a generative model for a pm 1q-player Markov game G with |G| ⁄ 2m^6, solves the pT;{p10mq;Nq-SPARSECCE problem for G for some T satisfying 1 T exppcmq, and any N P N. Then B must make at least 2 ^{pmq} queries to the generative model.

Theorem 5.2 establishes that there are m-player Markov games, where the number of states, actions per player, and horizon are bounded by polypmq, but any algorithm with regret opT{mq must make 2 ^{pmq} queries (via Fact C.1). In particular, if there are polypmq queries per episode, as is standard in the online simulator model where a trajectory is drawn from the policy ^{pt_q} at each episode t P rTs, then T ¡ 2 ^{pmq} episodes are required to have regret opT{mq. This is in stark contrast to the setting of normal-form games, where even for the case of bandit feedback (which is a special case of the generative model setting), stan-dard no-regret algorithms have the property that each player's regret scales as Op Tnq (i.e.,´ independently of m), where n de-notes the number of actions per player (Lattimore & Szepesvari, 2020). As with our computational lower bounds, Theorem 5.2 is not limited to decentralized algorithms, and also rules out cen-tralized algorithms which have access to a generative model.

Acknowledgements

This work was performed in part while Noah Golowich was an intern at Microsoft Research. Noah Golowich is supported at MIT by a Fannie & John Hertz Foundation Fellowship and an NSF Graduate Fellowship. This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence. Sham Kakade acknowledges funding from the Office of Naval Research under award N00014-22-1-2377 and the National Science Foundation Grant under award #CCF-2212841.

References

Abbasi Yadkori, Y., Bartlett, P. L., Kanade, V., Seldin, Y., and Szepesvari, C. Online learning in markov decision processes with adversarially chosen transition probability distributions. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. (eds.), Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/4f284803bd0966cc24fa8683a34afc6e-Paper.pdf.

Agapiou, J. P., Vezhnevets, A. S., Duéñez-Guzmán, E. A., Matyas, J., Mao, Y., Sunehag, P., Köster, R., Madhushani, U., Kopparapu, K., Comanescu, R., Strouse, D., Johanson, M. B., Singh, S., Haas, J., Mordatch, I., Mobbs, D., and Leibo, J. Z. Melting pot 2.0, 2022. URL https://arxiv.org/abs/2211.13746.

Anagnostides, I., Farina, G., Kroer, C., Lee, C.-W., Luo, H., and Sandholm, T. Uncoupled learning dynamics with $o(nlog t)$ swap regret in multiplayer games. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=CZwh1XdAhNv.

Arora, S. and Barak, B. Computational Complexity: A Modern Approach. Cambridge University Press, 2006. ISBN 978-0-521-42426-4.

Babichenko, Y. Query complexity of approximate nash equilibria. J. ACM, 63(4), oct 2016. ISSN 0004-5411.

Babichenko, Y. and Rubinstein, A. Communication complexity of approximate nash equilibria. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, pp. 878–889, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450345286. doi: 10.1145/3055399.3055407. URL https://doi.org/10.1145/3055399.3055407.

Bai, Y., Jin, C., and Yu, T. Near-optimal reinforcement learning with self-play. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., Jacob, A. P., Komeili, M., Konath, K., Kwon, M., Lerer, A., Lewis, M., Miller, A. H., Mitts, S., Renduchintala, A., Roller, S., Rowe, D., Shi, W., Spisak, J., Wei, A., Wu, D., Zhang, H., and Zijlstra, M. Human-level play in the game of ¡i¿diplomacy¡/i¿ by combining language models with strategic reasoning. Science, 378(6624):1067–1074, 2022. doi: 10.1126/science.ade9097. URL https://www.science.org/doi/abs/10.1126/science.ade9097.

Blackwell, D. An analog of the minimax theorem for vector payoffs. Pacific Journal of Mathematics, 6:1–8, 1956.

Borgs, C., Chayes, J., Immorlica, N., Kalai, A. T., Mirrokni, V., and Papadimitriou, C. The myth of the folk theorem. In Proceedings of the fortieth annual ACM symposium on Theory of computing, pp. 365–372, 2008.

Brown, G. W. Some notes on computation of games solutions. 1949.

Brown, N. and Sandholm, T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. Science, 359(6374):418–424, 2018. doi: 10.1126/science.aao1733. URL https://www.science.org/doi/abs/10.1126/science.aao1733.

Cesa-Bianchi, N. and Lugosi, G. Prediction, Learning, and Games. Cambridge University Press, New York, NY, USA, 2006. ISBN 0521841089.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., and Warmuth, M. K. How to use expert advice. Journal of the ACM, 44(3):427–485, 1997.

Chen, X. and Peng, B. Hedging in games: Faster convergence of external and swap regrets. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 18990–18999. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/db346ccb62d491029b590bbbf0f5c412-Paper.pdf.

Chen, X., Deng, X., and Teng, S.-h. Computing nash equilibria: Approximation and smoothed complexity. In 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pp. 603–612, 2006.

Chen, X., Cheng, Y., and Tang, B. Well-Supported vs. Approximate Nash Equilibria: Query Complexity of Large Games. In Papadimitriou, C. H. (ed.), 8th Innovations in Theoretical Computer Science Conference (ITCS 2017), volume 67 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 57:1–57:9, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-029-3. doi: 10.4230/LIPIcs.ITCS.2017.57. URL http://drops.dagstuhl.de/opus/volltexte/2017/8163.

Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. The complexity of computing a nash equilibrium. SIAM Journal on Computing, 39(1):195–259, 2009.

Daskalakis, C., Golowich, N., and Zhang, K. The complexity of markov equilibrium in stochastic games, 2022. URL https://arxiv.org/abs/2204.03991.

Daskalakis, C. C., Fishelson, M., and Golowich, N. Near-optimal no-regret learning in general

games. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), Advances in Neu-ral Information Processing Systems, 2021. URL https://openreview.net/forum?id=cVwc7IHWEWi.

Erez, L., Lancewicki, T., Sherman, U., Koren, T., and Mansour, Y. Regret minimization and convergence to equilibria in general-sum markov games, 2022. URL https://arxiv.org/abs/2207.14211.

Fearnley, J., Gairing, M., Goldberg, P., and Savani, R. Learning equilibria of games via payoff queries. In Proceedings of the Fourteenth ACM Conference on Electronic Commerce, EC '13, pp. 397–414, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319621. doi: 10.1145/2492002.2482558. URL https://doi.org/10.1145/2492002.2482558.

Foster, D. J., Rakhlin, A., Sekhari, A., and Sridharan, K. On the complexity of adversarial decision making. arXiv preprint arXiv:2206.13063, 2022.

Fudenberg, D., Levine, D., and Maskin, E. The folk theorem with imperfect public information. Econometrica, 62(5): 997–1039, 1994. ISSN 00129682, 14680262.

Hannan, J. Approximation to Bayes risk in repeated play. Contributions to the Theory of Games, 3:97–139, 1957.

Hart, S. and Mas-Colell, A. A simple adaptive procedure leading to correlated equilibrium. Econometrica, 68(5):1127–1150, 2000. doi: https://doi.org/10.1111/1468-0262.00153. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-0262.00153.

Jin, C., Krishnamurthy, A., Simchowitz, M., and Yu, T. Reward-free exploration for reinforcement learning. In International Conference on Machine Learning (ICML), pp. 4870–4879. PMLR, 2020.

Jin, C., Liu, Q., Wang, Y., and Yu, T. V-learning–a simple, efficient, decentralized algorithm for multiagent RL. arXiv preprint arXiv:2110.14555, 2021.

Jin, Y., Muthukumar, V., and Sidford, A. The complexity of infinite-horizon general-sum stochastic games, 2022.

Kakade, S. M. On the sample complexity of reinforcement learning, 2003.

Kearns, M., Mansour, Y., and Ng, A. Y. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'99, pp. 1324–1331, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Kramár, J., Eccles, T., Gemp, I., Tacchetti, A., McKee, K. R., Malinowski, M., Graepel, T., and Bachrach, Y. Negotiation and honesty in artificial intelligence methods for the board game of Diplomacy. Nature Communications, 13(1):7214, December 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-34473-5. URL https://www.nature.com/articles/s41467-022-34473-5. Number: 1 Publisher: Nature Publishing Group.

Kwon, J., Efroni, Y., Caramanis, C., and Mannor, S. RL for latent MDPs: Regret guarantees and a lower bound. Advances in Neural Information Processing Systems, 34, 2021.

Lattimore, T. and Szepesvári, C. Bandit algorithms. Cambridge University Press, 2020.

Leibo, J. Z., Dueñez-Guzman, E. A., Vezhnevets, A., Agapiou, J. P., Sunehag, P., Koster, R., Matyas, J., Beattie, C., Mordatch, I., and Graepel, T. Scalable evaluation of multi-agent reinforcement learning with melting pot. In Meila, M. and Zhang, T. (eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 6187–6199. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/leibo21a.html.

Littlestone, N. and Warmuth, M. K. The weighted majority algorithm. Information and computation, 108(2):212–261, 1994.

Littman, M. L. and Stone, P. A polynomial-time Nash equilibrium algorithm for repeated games. Decision Support Systems, 39:55–66, 2005.

Liu, Q., Wang, Y., and Jin, C. Learning Markov games with adversarial opponents: Efficient algorithms and fundamental limits. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pp. 14036–14053. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/liu22r.html.

Malialis, K. and Kudenko, D. Distributed response to network intrusions using multiagent reinforcement learning. Engineering Applications of Artificial Intelligence, 41:270–284, 2015. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2015.01.013. URL https://www.sciencedirect.com/science/article/pii/S095219761500024X.

Mao, W. and Basar, T. Provably efficient reinforcement learning in decentralized general-sum markov games. CoRR, abs/2110.05682, 2021. URL https://arxiv.org/abs/2110.05682.

Maskin, E. and Fudenberg, D. The folk theorem in repeated games with discounting or with incomplete information.

Econometrica, 53(3):533–554, 1986. Reprinted in A. Rubinstein (ed.), Game Theory in Economics, London: Edward Elgar, 1995. Also reprinted in D. Fudenberg and D. Levine (eds.), A Long-Run Collaboration on Games with Long-Run Patient Players, World Scientific Publishers, 2009, pp. 209-230.

Nash, J. Non-cooperative games. Annals of Mathematics, 54 (2):286–295, 1951. ISSN 0003486X.

Papadimitriou, C. H. On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci., 48(3):498–532, 1994.

Perolat, J., Vylder, B. D., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., McAleer, S., Elie, R., Cen, S. H., Wang, Z., Gruslys, A., Malysheva, A., Khan, M., Ozair, S., Timbers, F., Pohlen, T., Eccles, T., Rowland, M., Lanctot, M., Lespiau, J.-B., Piot, B., Omidshafiei, S., Lockhart, E., Sifre, L., Beauguerlange, N., Munos, R., Silver, D., Singh, S., Hassabis, D., and Tuyls, K. Mastering the game of stratego with model-free multiagent reinforcement learning. Science, 378(6623):990–996, 2022. doi: 10.1126/science.add4679. URL https://www.science.org/doi/abs/10.1126/science.add4679.

Puterman, M. Markov Decision Processes. John Wiley & Sons, Ltd, 1 edition, 1994. URL http://onlinelibrary.wiley.com/doi/10.1002/9780470316887.

Roughgarden, T. Intrinsic robustness of the price of anarchy. Journal of the ACM, 2015.

Rubinstein, A. Settling the complexity of computing approximate two-player Nash equilibria. In Annual Symposium on Foundations of Computer Science (FOCS), pp. 258–265. IEEE, 2016.

Rubinstein, A. Inapproximability of Nash equilibrium. SIAM Journal on Computing, 47(3):917–959, 2018.

Shalev-Shwartz, S., Shammah, S., and Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. CoRR, abs/1610.03295, 2016. URL http://arxiv.org/abs/1610.03295.

Shapley, L. Stochastic Games. PNAS, 1953.

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

Song, Z., Mei, S., and Bai, Y. When can we learn general-sum markov games with a large number of players sample-efficiently? In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=6MmiS0HUJHR.

Syrgkanis, V., Agarwal, A., Luo, H., and Schapire, R. E. Fast convergence of regularized learning in games. In Advances in Neural Information Processing Systems (NIPS), pp. 2989–2997, 2015.

Vovk, V. Aggregating strategies. Proc. of Computational Learning Theory, 1990, 1990.

Weisz, G., Amortila, P., Janzer, B., Abbasi-Yadkori, Y., Jiang, N., and Szepesvari, C. On query-efficient planning in mdps under linear realizability of the optimal state-value function. In Belkin, M. and Kpotufe, S. (eds.), Proceedings of Thirty Fourth Conference on Learning Theory, volume 134 of Proceedings of Machine Learning Research, pp. 4355–4385. PMLR, 15–19 Aug 2021.

Yin, D., Hao, B., Abbasi-Yadkori, Y., Lazić, N., and Szepesvári, C. Efficient local planning with linear function approximation. In Dasgupta, S. and Haghtalab, N. (eds.), Proceedings of The 33rd International Conference on Algorithmic Learning Theory, volume 167 of Proceedings of Machine Learning Research, pp. 1165–1192. PMLR, 29 Mar–01 Apr 2022.

Zhan, W., Lee, J. D., and Yang, Z. Decentralized optimistic hyperpolicy mirror descent: Provably no-regret learning in markov games. arXiv preprint arXiv:2206.01588, 2022.

Zheng, S., Trott, A., Srinivasa, S., Parkes, D. C., and Socher, R. The ai economist: Taxation policy design via two-level deep multiagent reinforcement learning. Science Advances, 8(18):eabk2607, 2022. doi: 10.1126/sciadv.abk2607. URL https://www.science.org/doi/abs/10.1126/sciadv.abk2607.

Contents of Appendix

Part I

Additional results and discussion

A. Tighter computational lower bounds under ETH for **PPAD**

Recall that Corollary 3.3 states that if PPAD P, then there is no constant C ¡ 4 and polyp|G|q-time algorithm which solves the p|G|^C;|G|$^{1\{C}$q-SPARSEMARKOVCCE problem for any 2-player Markov game G. Using a stronger complexity-theoretic assumption, the Exponential Time Hypothesis for PPAD (Rubinstein, 2016), we can obtain a hardness result which rules out

efficient algorithms even when 1) the accuracy ε is constant, as opposed to being $|G|^{-1/C}$, and 2) T is quasipolynomially large, as opposed to only being of polynomial size, i.e., $|G|^{C}$.

Corollary A.1 (ETH-hardness of SPARSEMARKOVCCE). There is a constant $\varepsilon_0 > 0$ such that if there exists an algorithm that solves the $(|G|^{\varepsilon_0 \log |G|}, \varepsilon_0)$-SPARSEMARKOVCCE problem in $|G|^{\varepsilon_0 \log |G|}$ time, then the Exponential Time Hypothesis for PPAD fails to hold.

Corollary A.1 is an immediate consequence of Theorem 3.2 and the fact that for some absolute constant $\varepsilon_0 > 0$, there are no polynomial-time algorithms for computing ε_0-Nash equilibria in 2-player normal-form games under the Exponential Time Hypothesis for PPAD (as shown in (Rubinstein, 2016)).

B. Discussion and interpretation

Theorems 3.2, 4.3, and 5.2 present barriers—both computational and statistical—toward developing efficient decentralized no-regret guarantees for multi-agent reinforcement learning. We emphasize that no-regret algorithms are the only known approach for obtaining fully decentralized learning algorithms (i.e., those which do not rely even on shared randomness) in normal-form games, and it seems unlikely that a substantially different approach would work in Markov games. Thus, these lower bounds for finding subexponential-length sequences of policies with the no-regret property represent a significant obstacle for fully decentralized multi-agent reinforcement learning. Moreover, these results rule out even the prospect of developing efficient centralized algorithms that produce no-regret sequences of policies, i.e., those which "resemble" independent learning. In this section, we compare our lower bounds with recent upper bounds for decentralized learning in Markov games, and explain how to reconcile these results.

B.1. Comparison to V-learning

The V-learning algorithm (Jin et al., 2021; Song et al., 2022; Mao & Basar, 2021) is a polynomial-time decentralized learning algorithm that proceeds in two phases. In the first phase, the m agents interact over the course of K episodes in a decentralized fashion, playing product Markov policies $\pi^{(1)}, \ldots, \pi^{(K)} \in \Pi^{\text{markov}}$. In the second phase, the agents use data gathered during the first phase to produce a distributional policy $\mu \in \Delta(\Pi^{\text{gen;rnd}})$, which we refer to as the output policy of V-learning. As discussed in Section 1, one implication of Theorem 3.2 is that the first phase of V-learning cannot guarantee each agent sublinear regret. Indeed if K is of polynomial size (and PPAD $\neq$ P), this follows because a bound of the form $\mathrm{Reg}_{i,K}(\pi^{(1)}, \ldots, \pi^{(K)}) / K$ for all i implies that $(\pi^{(1)}, \ldots, \pi^{(K)})$ solves the (K, ε)-SPARSEMARKOVCCE problem.

The output policy $\mu \in \Delta(\Pi^{\text{gen;rnd}})$ produced by V-learning is an approximate CCE (per Definition 2.1), and it is natural to ask how many product policies it takes to represent μ as a uniform mixture (that is, whether μ solves the (ε, T)-SPARSEMARKOVCCE problem for a reasonable value of T). First, recall that V-learning requires $K = \mathrm{poly}(H, S, \max_i A_i, 1/\varepsilon)^2$ episodes to ensure that μ is an ε-CCE. It is straightforward to show that μ can be expressed as a non-uniform mixture of at most $K^{KHS} + 1$ policies in $\Pi^{\text{gen;rnd}}$ (we prove this fact in detail below). By discretizing the non-uniform mixture, one can equivalently represent it as uniform mixture of $O(1/\varepsilon)^{KHS} + 1$ product policies, up to ε error. Recalling the value of K, we conclude that we can express μ as a uniform mixture of $T = \exp(O(1/\varepsilon)^2 \cdot \mathrm{poly}(H, S, \max_i A_i))$ product policies in $\Pi^{\text{gen;rnd}}$. Note that the lower bound of Theorem 4.3 rules out the efficient computation of an ε-CCE represented as a uniform mixture of $T = \exp(\varepsilon^2 \max(H, S, \max_i A_i))$ efficiently computable policies in $\Pi^{\text{gen;rnd}}$. Thus, in the regime where $1/\varepsilon$ is polynomial in $H, S, \max_i A_i$, this upper bound on the sparsity of the policy μ produced V-learning matches that from Theorem 4.3, up to a polynomial in the exponent.

The sparsity of the output policy from V-learning. We now sketch a proof of the fact that the output policy μ produced by V-learning can be expressed as a (non-uniform) average of $K^{KHS} + 1$ policies in $\Pi^{\text{gen;rnd}}$, where K is the number of episodes in the algorithm's initial phase. We adopt the notation and terminology from Jin et al. (2021).

Consider Algorithm 3 of Jin et al. (2021), which describes the second phase of V-learning, which produces the output policy μ. We describe how to write μ as a weighted average of a collection of product policies, each of which is indexed by a function $\nu : [H] \times S \times [K] \to [K]$ and a parameter $k_0 \in [K]$: in particular, we will write $\mu = \sum_{k_0, \nu} w_{k_0, \nu} \pi_{k_0, \nu}$, where $w_{k_0, \nu} \in [0, 1]$ are mixing weights summing to 1 and $\pi_{k_0, \nu} \in \Pi^{\text{gen;rnd}}$. The number of tuples (k_0, ν) is $K^{1 + KHS}$.

We define the mixing weight allocated $w_{k_0, \nu}$ to any tuple (k_0, ν) to be:

$$\frac{1}{K} \prod_{(h, s, k) \in [H] \times [S] \times [K]} \mathbb{1}(\nu(h, s, k) = N_h^k(s, k)),$$

where $N_h^k(s, k) \in [K]$ and $N_h^{\nu(h,s,k)}(s, k) \in [0, 1]$ (for $i \in [N_h(s)]$) are defined as in (Jin et al., 2021).

Next, for each k_0;, we define $_{k\ ;}$ $P_0^{gen;rnd}$ to be the following policy: it maintains a parameter k P rKs over the first h ∕ H steps of the episode (as in Algorithm 3 of (Jin et al., 2021)), but upon reaching state s at step h, given the present value of k P rKs, sets i:ph;s;kq, and updates k — k^i psq, and then samples an action a^kp|sq (where $_h k^i$ psq; ^{k}p|sq are defined in (Jin et al., 2021)). Since the mixing weights $w_{k\ ;}$ defined above exactly simulate the random draws of the parameter k in Line 1 and the parameters i in Algorithm 3, Line 4 of (Jin et al., 2021), it follows that the distributional policy defined by Algorithm 3 of (Jin et al., 2021) is equal to $_{k_0;}$ $w_{k_0;k_0;}$ Pp$^{gen;rnd}$q.

B.2. No-regret learning against Markov deviations

As discussed in Section 1, Erez et al. (2022) showed the existence of a learning algorithm with the property that if each agent plays it independently for T episodes, then no player can achieve regret more than Oppolypm;H;S;max$_i$A$_i$qT$^{3\{4}$q by deviating to any fixed Markov policy. This notion of regret corresponds to, in the context of Definition 2.2, replacing max $_{P^{gen;rnd}}$ with the smaller quantity max $_{P_i^{markov}}$. Thus, the result of Erez et al. (2022) applies to a weaker notion of regret than that of the SPARSECCE problem, and so does not contradict any of our lower bounds. One may wonder which of these two notions of regret (namely, best possible gain via deviation to a Markov versus non-Markov policy) is the "right" one. We do not believe that there is a definitive answer to this question, but we remark that in many empirical applications of multi-agent reinforcement learning it is standard to consider non-Markov policies (Leibo et al., 2021; Agapiou et al., 2022). Furthermore, as shown in the proposition below, there are extremely simple games, e.g., of constant size, in which Markov deviations lead to "vacuous" behavior: in particular, all Markov policies have the same (suboptimal) value but the best non-Markov policy has much greater value:

Proposition B.1. There is a 2-player, 2-action, 1-state Markov game with horizon 2 and a non-Markov policy $P_2^{gen;rnd}$ for player 2 so that for all $_1$ P^{markov}, $V^{1\,2}$ $1\{2$ yet max$_{1\ P^{gen;rnd}}$ $V^{1\,2}$ $3\{4$.

The proof of Proposition B.1 is provided in Section B.5 below.

Other recent work has also proved no-regret guarantees with respect to deviations to restricted policy classes. In particular, Zhan et al. (2022) studies a setting in which each agent i is allowed to play policies in an arbitrary restricted policy class $_i$ ^{1} $_i$, $^{gen;rnd}$ in each episode, and regret is measured with respect to deviations to any policy in ^1_i. Zhan et al. (2022) introduces an algorithm, DORIS, with the property that when all agents play it independently, each agent i experiences regret O polypm;A;S;Hq $\sqrt{T \sum_{i=1}^{m} \log|^1|}$ to their respective class ^{1}.[15]

DORIS is not computationally efficient, since it involves performing exponential weights over the class ^{1}, which requires space complexity j^{1}j. Nonetheless, one can compare the statistical guarantees the algorithm provides to our own results. Let $^{markov;det}$ $\in^{markov}$ denote the set of deterministic Markov policies of agent i, namely sequences $_i$ p$_{i;1}$;:::;$_{i;H}$q so that $_{i;h}$: S Ñ A$_i$. In the case that ^{1} $^{markov;det}$, ^{1}, we have log|^1_i| OpSHlogA$_i$q, which means that DORIS obtains no-regret against Markov deviations when m is constant, comparable to Erez et al. (2022).[16] However, we are interested in the setting in which each player's regret is measured with respect to all deviations in $^{gen;rnd}$ (equivalently, $^{gen;det}$). Accordingly, if we take ^{1} $^{gen;det}$ $\in^{gen;rnd}$,[17] then log|^1| pSA$_i$q^{H1}, meaning that DORIS does not imply any sort of sample-efficient guarantee, even for m 2.

Finally, we remark that the algorithm DORIS (Zhan et al., 2022), as well as the similar algorithm OPMD from earlier work of Liu et al. (2022), obtains the same regret bound stated above even when the opponents are controlled by (possibly adaptive) adversaries. However, this guarantee crucially relies on the fact that any agent implementing DORIS must observe the policies played by opponents following each episode; this feature is the reason that the regret bound of DORIS does not contradict the exponential lower bound of Liu et al. (2022) for no-regret learning against an adversarial opponent. As a result of being restricted to this "revealed-policy" setting, DORIS is not a fully decentralized algorithm in the sense we consider in this paper.

[15] Note that in the tabular setting, the sample complexity of DORIS (Corollary 1) scales with the size A of the joint action set, since each player's value function class consists of the class of all functions f : S A Ñ r0;1s, which has Eluder dimension scaling with S A, i.e., exponential in m.

[16] Erez et al. (2022) has the added bonus of computational efficiency, even for polynomially large m, though has the significant drawback of assuming that the Markov game is known.

[17] DORIS plays distributions over policies in ^{1} $^{gen;det}$ at each episode, whereas in our lower bounds we consider the setting where a policy in $^{gen;rnd}$ is played each episode; Facts F.2 and F.3 shows that these two settings are essentially equivalent, in that any policy in $^{gen;rnd}$ can be simulated by one in p$^{gen;det}$q and vise versa.

B.3. On the role of shared randomness

A key assumption in our lower bounds for no-regret learning is that each of the joint policies $\{\pi^1\},\ldots,\{\pi^T\}$ produced by the algorithm is a product policy; such an assumption is natural, since it subsumes independent learning protocols in which each agent i selects $\{\pi^t\}_i$ without knowledge of $\{\pi^t\}_{-i}$. Compared to general (stochastic) joint policies, product policies have the desirable property that, to sample a trajectory from $\{\pi^t\}\sim\mathsf{P}^{\mathsf{gen;rnd}}_1,\ldots,\{\pi^t\}\sim\mathsf{P}^{\mathsf{gen;rnd}}_m$, the agents do not require access to shared randomness. In particular, each agent i can independently sample its action from $\{\pi^t\}_i\sim\mathsf{P}^{\mathsf{gen;rnd}}_i$ at each of the H steps of the episode. It is natural to ask how the situation changes if we allow the agents to use shared random bits when sampling from their policies, which corresponds to allowing $\{\pi^1\},\ldots,\{\pi^T\}$ to be non-product policies. In this case, V-learning yields a positive result via a standard "batch-to-online" conversion: by applying the first phase of V-learning during the first $T^{2/3}$ episodes and playing trajectories sampled i.i.d. from the output policy produced by V-learning during the remaining $T-T^{2/3}$ episodes (which requires shared randomness), it is straightforward to see that a regret bound of order $\mathrm{poly}(H,S,\max_i A_i)T^{2/3}$ can be obtained. Similar remarks apply to SPoCMAR (Daskalakis et al., 2022), which can obtain a slightly worse regret bound of order $\mathrm{poly}(H,S,\max_i A_i)T^{3/4}$ in the same fashion. In fact, the batch-to-online conversion approach gives a generic solution for the setting in which shared randomness is available. That is, the assumption of shared randomness eliminates any distinction between no-regret algorithms and (non-sparse) equilibrium computation algorithms, modulo slight loss in rates. For this reason, the shared randomness assumption is too strong to develop any sort of distinct theory of no-regret learning.

B.4. Comparison to lower bounds for finding stationary CCE

A separate line of work Daskalakis et al. (2022); Jin et al. (2022) has recently shown PPAD-hardness for the problem of finding stationary Markov CCE in infinite-horizon discounted stochastic games. These results are incomparable with our own: stationary Markov CCE are not sparse (in the sense of Definition 3.1), whereas we do not require stationarity of policies (as is standard in the finite-horizon setting).

B.5. Proof of Proposition B.1

Below we prove Proposition B.1.

Proof of Proposition B.1. We construct the claimed Markov game G as follows. The single state is denoted by s; as there is only a single state, the transitions are trivial. We denote each player's action space as $\mathcal{A}_1=\mathcal{A}_2=\{1,2\}$. The rewards to player 1 are given as follows: for all $(a_1,a_2)\in\mathcal{A}$,

$$R_{1,1}(s,(a_1,a_2))=\mathbb{I}_{\{a_2=1\}},\qquad R_{1,2}(s,(a_1,a_2))=\mathbb{I}_{\{a_1=a_2\}}.$$

We allow the rewards of player 2 to be arbitrary; they do not affect the proof in any way.

We let $\pi_2=(\pi_{2,1},\pi_{2,2},\ldots)\sim\mathsf{P}^{\mathsf{gen;rnd}}_2$ be the policy which plays a uniformly random action at step 1 and then plays the same action at step 2: formally, $\pi_{2,1}(s_1)=\mathrm{Unif}(\mathcal{A}_2)$, and $\pi_{2,2}(s_1,a_{2,1},r_{2,1},s_2)=\mathbb{I}_{\{a_{2,1}\}}$. Then for any Markov policy $\pi_1\in\mathsf{P}^{\mathsf{markov}}_1$ of player 1, we must have $\mathbb{P}_{\pi_1\times\pi_2}(a_{1,2}=a_{2,2})=1/2$, which means that $V_1^{\pi_1\times\pi_2}=\frac{1}{2}\mathbb{E}_{\pi_1\times\pi_2}[\mathbb{I}_{\{a_{2,1}=1\}}+\mathbb{I}_{\{a_{1,2}=a_{2,2}\}}]=1/2+1/2=1$.

On the other hand, any general (non-Markov) policy $\pi_1\in\mathsf{P}^{\mathsf{gen;rnd}}_1$ which satisfies

$$\pi_{1,2}(\cdot\mid(s_1,a_{1,1},r_{1,1},s_2))=\begin{cases}\mathcal{I}_1: & r_{1,1}=1\\ \mathcal{I}_2: & r_{1,1}=0\end{cases}$$

has $V_1^{\pi_1\times\pi_2}=1/2\cdot(1/2+1)=3/4$. □

Part II

Proofs

C. Additional preliminaries

C.1. Additional preliminaries for Markov games

Deterministic policies. It will be helpful to introduce notation for deterministic general (non-Markov) policies, which correspond to the special case of randomized policies where each policy $\pi_{i;h}$ exclusively maps to singleton distributions. In particular, a deterministic general policy of agent i is a collection of mappings $\pi_i = (\pi_{i;1}, \ldots, \pi_{i;H})$, where $\pi_{i;h} : \mathcal{H}_{i;h-1} \to \mathcal{A}_i$. We denote by $\Pi_i^{\mathrm{gen;det}}$ the space of deterministic general policies of agent i, and further write $\Pi^{\mathrm{gen;det}} := \Pi_1^{\mathrm{gen;det}} \times \cdots \times \Pi_m^{\mathrm{gen;det}}$ to denote the space of joint deterministic policies. We use the convention throughout that deterministic policies are denoted by the letter π, whereas randomized policies are denoted by σ.

Additional facts on regret and CCE. The following facts regarding deterministic policies and the definition of coarse correlated equilibria and regret are well-known:

- In the context of Definition 2.1 (defining an ε-CCE), the maximizing policy π_i' can always be chosen to be deterministic, so $\sigma \in \Pi^{\mathrm{gen;rnd}}$ is an ε-CCE if and only if $\max_{i} \max_{\pi_i' \in \Pi_i^{\mathrm{gen;det}}} V_i^{\pi_i', \sigma_{-i}} - V_i^{\sigma} \le \varepsilon$.
- In the context of (1) in the definition of regret, the maximum over $\pi_i \in \Pi_i^{\mathrm{gen;rnd}}$ is always achieved by a deterministic general policy, so we have $\mathrm{Reg}_{i;T} = \max_{\pi_i \in \Pi_i^{\mathrm{gen;det}}} \sum_{t=1}^{T} V_{i}^{\pi_i, \sigma_{-i}^{(t)}} - V_i^{\sigma^{(t)}}$.

Next, the following standard result shows that the uniform average of any no-regret sequence forms an approximate coarse correlated equilibrium.

Fact C.1 (No-regret is equivalent to CCE). Suppose that a sequence of policies $\sigma^{(1)}, \ldots, \sigma^{(T)} \in \Pi^{\mathrm{gen;rnd}}$ satisfies $\mathrm{Reg}_{i;T}(\sigma^{(1)}, \ldots, \sigma^{(T)}) \le \varepsilon \cdot T$ for each $i \in [m]$. Then the uniform average of these T policies, namely the distributional policy $\bar{\sigma} := \frac{1}{T} \sum_{t=1}^{T} \delta_{\sigma^{(t)}} \in \Delta(\Pi^{\mathrm{gen;rnd}})$, is an ε-CCE.

Likewise if a sequence of policies $\sigma^{(1)}, \ldots, \sigma^{(T)} \in \Pi^{\mathrm{gen;rnd}}$ has the property that the distributional policy $\bar{\sigma} := \frac{1}{T} \sum_{t=1}^{T} \delta_{\sigma^{(t)}} \in \Delta(\Pi^{\mathrm{gen;rnd}})$, is an ε-CCE, then we have $\mathrm{Reg}_{i;T}(\sigma^{(1)}, \ldots, \sigma^{(T)}) \le \varepsilon T$ for all $i \in [m]$.

Fact C.1 is an immediate consequence of Definitions 2.1 and 2.2.

C.2. Nash equilibria and computational hardness.

The most foundational and well known solution concept for normal-form games is the Nash equilibrium (Nash, 1951).

Definition C.2 ((n, q)-NASH problem). For a normal-form game $G = (M_1, \ldots, M_m)$ and $\varepsilon \ge 0$, a product distribution $p = \prod_{j=1}^{m} p_j \in \Delta([n])^{\otimes m}$ is said to be an ε-Nash equilibrium for G if for all $i \in [m]$,

$$\max_{a_i' \in [n]} \mathbb{E}_{a \sim p}\left[M_i(a_i', a_{-i})\right] - \mathbb{E}_{a \sim p}\left[M_i(a)\right] \le \varepsilon.$$

We define the m-player (n, q)-NASH problem to be the problem of computing an ε-Nash equilibrium of a given m-player n-action normal-form game.[18]

Informally, p is an ε-Nash equilibrium if no player i can gain more than ε in reward by deviating to a single fixed action a_i', while all other players randomly choose their actions according to p. Despite the intuitive appeal of Nash equilibria, they are intractable to compute: for any $c > 0$, it is PPAD-hard to solve the (n, n^{-c})-NASH problem, namely, to compute n^{-c}-approximate Nash

[18] One must also take care to specify the bit complexity of representing a normal-form game. We assume that the payoffs of any normal-form game given as an instance to the (n, q)-NASH problem can each be expressed with $\max(n, m)$ bits; this assumption is without loss of generality as long as $q \ge 2^{-\max(n,m)}$ (which it will be for us).

equilibria in 2-player n-action normal-form games (Daskalakis et al., 2009; Chen et al., 2006; Rubinstein, 2018). We recall that the complexity class PPAD consists of all total search problems which have a polynomial-time reduction to the End-of-The-Line (EOTL) problem. PPAD is the most well-studied complexity class in algorithmic game theory, and it is widely believed that PPAD $\neq$ P. We refer the reader to (Daskalakis et al., 2009; Chen et al., 2006; Rubinstein, 2018; Papadimitriou, 1994) for further background on the class PPAD and the EOTL problem.

C.3. Query complexity of Nash equilibria

Our statistical lower bound for the SPARSECCE problem in Theorem 5.2 relies on existing query complexity lower bounds for computing approximate Nash equilibria in m-player normal-form games. We first review the query complexity model for normal-form games.

Oracle model for normal-form games. For $m, n \in \mathbb{N}$, consider an m-player n-action normal form game G, specified by payoff tensors $M_1, \ldots, M_m$. Since the tensors $M_1, \ldots, M_m$ contain a total of mn^m real-valued payoffs, in the setting when m is large, it is unrealistic to assume that an algorithm is given the full payoff tensors as input. Therefore, prior work on computing equilibria in such games has studied the setting in which the algorithm makes adaptive oracle queries to the payoff tensors.

In particular, the algorithm, which is allowed to be randomized, has access to a payoff oracle $\mathcal{O}_G$ for the game G, which works as follows. At each time step, the algorithm can choose to specify an action profile $\mathbf{a} \in [n]^m$ and then query $\mathcal{O}_G$ at the action profile $\mathbf{a}$. The oracle $\mathcal{O}_G$ then returns the payoffs $M_1[\mathbf{a}], \ldots, M_m[\mathbf{a}]$ for each player if the action profile $\mathbf{a}$ is played.

Query complexity lower bound for approximate Nash equilibrium. The following theorem gives a lower bound on the number of queries any randomized algorithm needs to make to compute an approximate Nash equilibrium in an m-player game.

Theorem C.3 (Corollary 4.5 of (Rubinstein, 2016)). There is a constant $\varepsilon_0 > 0$ so that any randomized algorithm which solves the $(2, \varepsilon_0)$-NASH problem for m-player normal-form games with probability at least $2/3$ must use at least $2^{\Omega(m)}$ payoff queries.

We remark that (Babichenko, 2016; Chen et al., 2017) provide similar, though quantitatively weaker, lower bounds to that in Theorem C.3. We also emphasize that the lower bound of Theorem C.3 applies to any algorithm, i.e., including those which require extremely large computation time.

D. Proofs of lower bounds for SPARSEMARKOVCCE (Section 3)

D.1. Preliminaries: Online density estimation

Our proof makes use of tools for online learning with the logarithmic loss, also known as conditional density estimation. In particular, we use a variant of the exponential weights algorithm known as Vovk's aggregating algorithm in the context of density estimation (Vovk, 1990; Cesa-Bianchi & Lugosi, 2006). We consider the following setting with two players, a Learner and Nature. Furthermore, there is a set $\mathcal{Y}$, called the outcome space, and a set $\mathcal{X}$, called the context space; for our applications it suffices to assume $\mathcal{Y}$ and $\mathcal{X}$ are finite. For some $T \in \mathbb{N}$, there are T time steps $t = 1, 2, \ldots, T$. At each time step $t \in [T]$:

- Nature reveals a context $x^{(t)} \in \mathcal{X}$;
- Having seen the context $x^{(t)}$, the learner predicts a distribution $\hat{p}^{(t)} \in \Delta(\mathcal{Y})$;
- Nature chooses an outcome $y^{(t)} \in \mathcal{Y}$, and the learner suffers loss $\ell_{\log}(\hat{p}^{(t)}, y^{(t)}) := \log \frac{1}{\hat{p}^{(t)}(y^{(t)})}$.

For each $t \in [T]$, we let $H^{(t)} = (x^{(1)}, y^{(1)}, \hat{p}^{(1)}, \ldots, x^{(t)}, y^{(t)}, \hat{p}^{(t)})$ denote the history of interaction up to step t; we emphasize that each context $x^{(t)}$ may be chosen adaptively as a function of $H^{(t-1)}$. Let $\mathcal{F}^{(t)}$ denote the sigma-algebra generated by $(H^{(t)}, x^{(t+1)})$. We measure performance in terms of regret against a set $\mathcal{I}$ of experts, also known as the expert setting. Each expert $i \in \mathcal{I}$ consists of a function $p_i : \mathcal{X} \to \Delta(\mathcal{Y})$. The regret of an algorithm against the expert class $\mathcal{I}$ when it receives contexts $x^{(1)}, \ldots, x^{(T)}$ and observes outcomes $y^{(1)}, \ldots, y^{(T)}$ is defined as

$$\mathrm{Reg}_{\mathcal{I},T} = \sum_{t=1}^{T} \ell_{\log}(\hat{p}^{(t)}, y^{(t)}) - \min_{i \in \mathcal{I}} \sum_{t=1}^{T} \ell_{\log}(p_i(x^{(t)}), y^{(t)}).$$

Note that the learner can observe the expert predictions $(p_i(x^{(t)}))_{i \in \mathcal{I}}$ and use them to make its own prediction at each round t.

Proposition D.1 (Vovk's aggregating algorithm). Consider Vovk's aggregating algorithm, which predicts via

$$\hat{p}^{(t)}(y) := E_{i\sim q^{(t)}}[p_i(x^{(t)})(y)], \quad \text{where} \quad q^{(t)}(i) := \frac{\exp\left(\eta \sum_{s=1}^{t-1} \log p_i(x^{(s)})(y^{(s)})\right)}{\sum_{j\in I} \exp\left(\eta \sum_{s=1}^{t-1} \log p_j(x^{(s)})(y^{(s)})\right)}. \tag{2}$$

This algorithm guarantees a regret bound of $\mathrm{Reg}_{I,T} \le \sqrt{\log|I|}$.

Recall that for probability distributions p, q on a finite set B, their total variation distance is defined as

$$D_{TV}(p, q) = \max_{E\subseteq B} |p(E) - q(E)|. \tag{3}$$

As a (standard) consequence of Proposition D.1, in the realizable setting in which the distribution of $y^{(t)}|x^{(t)}$ follows $p_i(x^{(t)})$ for some fixed (unknown) expert $i \in I$, we can obtain a bound on the total variation distance between the algorithm's predictions and those of $p_i(x^{(t)})$.

Proposition D.2. If the distribution of outcomes is realizable, i.e., there exists an expert $i \in I$ so that $y^{(t)} \sim p_i(x^{(t)}) \,|\, x^{(t)}, H^{(t-1)}$ for all $t \in [T]$, then the predictions $\hat{p}^{(t)}$ of the aggregation algorithm (2) satisfy

$$\sum_{t=1}^{T} \mathbb{E}\left[D_{TV}\left(\hat{p}^{(t)}, p_i(x^{(t)})\right)\right] \le \sqrt{T \log|I|}.$$

For completeness, we provide the proof of Proposition D.2 here.

Proof of Proposition D.2. To simplify notation, for an expert $i \in I$, a context $x \in X$, and an outcome $y \in Y$, we write $p_i(y|x)$ to denote $p_i(x)(y)$.

Proposition D.1 gives that the following inequality holds (almost surely):

$$\mathrm{Reg}_{I,T} \le \sum_{t=1}^{T} \log\frac{1}{q^{(t)}(y^{(t)})} - \sum_{t=1}^{T} \log\frac{1}{p_i(y^{(t)}|x^{(t)})} \Big/ \sqrt{\log|I|}.$$

For each $t \in [T]$, note that $\hat{p}^{(t)}$ and $x^{(t)}$ are $F^{(t-1)}$-measurable (by definition). Then

$$\begin{aligned}
\sum_{t=1}^{T} D_{TV}\left(\hat{p}^{(t)}, p_i(x^{(t)})\right)^2 &\le \sum_{t=1}^{T} D_{KL}\left(p_i(x^{(t)}) \,\|\, \hat{p}^{(t)}\right) \\
&= \sum_{t=1}^{T} \sum_{y\in Y} p_i(y|x^{(t)}) \log\frac{p_i(y|x^{(t)})}{\hat{p}^{(t)}(y)} \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\log\frac{1}{\hat{p}^{(t)}(y^{(t)})} - \log\frac{1}{p_i(y^{(t)}|x^{(t)})} \,\middle|\, F^{(t-1)}\right],
\end{aligned}$$

where the first inequality uses Pinsker's inequality and the final equality uses the fact that $y^{(t)} \sim p_i(x^{(t)})|x^{(t)}, H^{(t-1)}$. It follows that

$$\mathbb{E}\left[\sum_{t=1}^{T} D_{TV}\left(\hat{p}^{(t)}, p_i(x^{(t)})\right)^2\right] \le \mathbb{E}[\mathrm{Reg}_{I,T}] \sqrt{\log|I|}.$$

Jensen's inequality now gives that

$$\mathbb{E}\left[\sum_{t=1}^{T} D_{TV}\left(\hat{p}^{(t)}, p_i(x^{(t)})\right)\right] \le \sqrt{T \cdot \mathbb{E}\left[\sum_{t=1}^{T} D_{TV}\left(\hat{p}^{(t)}, p_i(x^{(t)})\right)^2\right]} \le \sqrt{T \log|I|}.$$

□

D.2. Proof of Theorem 3.2

Proof of Theorem 3.2. Fix $n \in \mathbb{N}$, which we recall represents an upper bound on the description length of the Markov game. Assume that we are given an algorithm $\mathcal{B}$ that solves the (T, ε)-SPARSEMARKOVCCE problem for Markov games G satisfying $|G| \le n$ in time U. We proceed to describe an algorithm which solves the 2-player $(n^{1/2}, 2u, 4\varepsilon)$-NASH problem in time $n^{T} U + C_0$, as long as $T \ge \exp(2 n^{1/2} \log^5 n)$. First, define $n_0 := n^{1/2} \log^2 n$, and consider an arbitrary 2-player n_0-action normal form G, which is specified by payoff matrices $M_1, M_2 \in [0,1]^{n_0 \times n_0}$, so that all entries of the game can be written in binary using at most n_0 bits (recall, per footnote 18, that we may assume that the entries of an instance of $(n_0, 4\varepsilon)$-NASH can be specified with n_0 bits). Based on G, we construct a 2-player Markov game $G' := G(G)$ as follows:

Definition D.3. We define the game $G(G)$ to consist of the tuple $G(G) = (S, H, (A_i)_{i \in [2]}, P, (R_i)_{i \in [2]}, \zeta)$, where:

- The horizon of G is $H = 2\lfloor n_0 \rfloor = 2u$ (i.e., the largest even number at most n_0).
- Let $A = n_0$; the action spaces of the 2 agents are given by $A_1 = A_2 = [A]$.
- There are a total of $A^2 + 1$ states: in particular, there is a state $s_{(a_1, a_2)}$ for each $(a_1, a_2) \in [A]^2$, as well as a distinguished state $s_\star$, so we have:
$$S = \{s_\star\} \cup \{s_{(a^1, a^2)} : (a_1, a_2) \in [A]^2\}.$$
- For all odd $h \in [H]$, the reward to agents $j \in [2]$ given that the action profile (a_1, a_2) is played at step h is given by $R_{j,h}(s_\star, (a_1, a_2)) = \frac{1}{H} M_j(a_1, a_2)$, for all $s \in S$. All agents receive 0 reward at even steps $h \in [H]$.
- At odd steps $h \in [H]$, if actions $a_1, a_2 \in [A]$ are taken, the game transitions to the state $s_{(a_1, a_2)}$. At even steps $h \in [H]$, the game always transitions to the state $s_\star$.
- The initial state (i.e., at step $h = 1$) is $s_\star$ (i.e., ζ is a singleton distribution supported on $s_\star$).

It is evident that this construction takes polynomial time, and satisfies $|G| / A^2 = 1/n^2 \le 1/n$. We will now show by applying the algorithm $\mathcal{B}$ to G, we can efficiently compute 4-approximate Nash equilibrium for the original game G. To do so, we appeal to Algorithm 1.

Algorithm 1 Algorithm to compute Nash equilibrium used in proof of Theorem 3.2.

1: Input: 2-player, n_0-action normal form game G.
2: Construct the 2-player Markov game $G' = G(G)$ per Definition D.3, which satisfies $|G'| \le n$.
3: Call the algorithm $\mathcal{B}$ on the game G', which produces a sequence $\pi^{(1)}, \ldots, \pi^{(T)}$, where each $\pi^{(t)} \in \Pi_{\mathrm{markov}}$.
4: for $t \in [T]$ and odd $h \in [H]$: do
5: if $\pi^{(t)}_h(s_\star) \in \Delta(A_1) \times \Delta(A_2)$ is a $(4\varepsilon, \eta)$-Nash equilibrium of G: then 6:
return $\pi^{(t)}_h(s_\star)$.
7: end if
8: end for
9: if the for loop terminates without returning: return fail.

Algorithm 1 proceeds as follows. First, it constructs the 2-player Markov game G' as defined above, and calls the algorithm $\mathcal{B}$, which returns a sequence $\pi^{(1)}, \ldots, \pi^{(T)} \in \Pi_{\mathrm{markov}}$ of product Markov policies with the property that the average $\frac{1}{T} \sum_{t=1}^{T} \pi^{(t)}$ is an ε-CCE of G'. It then enumerates over the distributions $\pi^{(t)}_h(s_\star) \in \Delta(A_1) \times \Delta(A_2)$ for each $t \in [T]$ and $h \in [H]$ odd, and checks whether each one is a 4-approximate Nash equilibrium of G. If so, the algorithm outputs such a Nash equilibrium, and otherwise, it fails. The proof of Theorem 3.2 is thus completed by the following lemma, which states that as long as ε is an ε-CCE of G', Algorithm 1 never fails.

Lemma D.4 (Correctness of Algorithm 1). Consider the normal form game G and the Markov game $G' = G(G)$ as constructed above, which has horizon H. For any $\varepsilon_0 > 0$, $T \in \mathbb{N}$, if $T \ge \exp(H^2 \log^8 n / \varepsilon_0^2)$ and $\pi^{(1)}, \ldots, \pi^{(T)} \in \Pi_{\mathrm{markov}}$ are product Markov policies so that $\frac{1}{T} \sum_{t=1}^{T} \pi^{(t)}$ is an $(\varepsilon_0 \log^4 n)$-CCE of G', then there is some odd $h \in [H]$ and $t \in [T]$ so that $\pi^{(t)}_h(s_\star)$ is an ε_0-Nash equilibrium of G.

The proof of Lemma D.4 is given below. Applying Lemma D.4 with $\varepsilon_0 = 4$ (which is a valid application since $T = \exp(n_0)(4q^2)\{(2^8 q)$ by our assumption on T;), yields that Algorithm 1 always finds a 4-Nash equilibrium of the n_0-action normal form game G, thus solving the given instance of the $(n_0;4q)$-NASH problem. Furthermore, it is straightforward to see that Algorithm 1 runs in time $O(nTq^{C_0}) = O(nTq^{C_0})$, for some constant $C_0 \geq 1$.

□

Proof of Lemma D.4. Consider a sequence of product Markov policies $\pi^{(1)}, \ldots, \pi^{(T)}$ with the property that the average $\frac{1}{T}\sum_{t=1}^{T} \pi^{(t)}$ is an $(\varepsilon_0/4)$-CCE of G. For all odd $h \in [H]$ and $j \in [2]$, let $\pi^{(t)}_{j,h} := \pi^{(t)}_h(s) \in \Delta(\mathcal{A}_j)$, which is the distribution played under $\pi^{(t)}$ by player j at step h (at the unique state s with positive probability of being reached at step h). For odd h, we have $\pi^{(t)}_h(s) = \pi^{(t)}_{1,h} \times \pi^{(t)}_{2,h}$, and our goal is to show that for some odd $h \in [H]$ and $t \in [T]$, $\pi^{(t)}_{1,h} \times \pi^{(t)}_{2,h}$ is an ε_0-Nash equilibrium of G. To proceed, suppose for the sake of contradiction that this is not the case.

Let us write $\mathcal{O}_H := \{h \in [H] : h \text{ odd}\}$ to denote the set of odd-numbered steps, and $\mathcal{E}_H = [H] \setminus \mathcal{O}_H$ to denote the set of even-numbered steps. Let $H_0 := |\mathcal{O}_H| \cdot |\mathcal{E}_H| = H/2$. We first note that for $j \in [2]$, agent j's value under the mixture policy $\frac{1}{T}\sum_{t=1}^{T} \pi^{(t)}$ is given as follows:

$$V_j = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{H} \sum_{h \in \mathcal{O}_H} \mathbb{E}_{(a^1, a^2) \sim \pi^{(t)}_{1,h} \times \pi^{(t)}_{2,h}} r(M_j, q_{(a^1, a^2)}).$$

For each $j \in [2]$, we will derive a contradiction by constructing a (non-Markov) deviation policy for player j in G, denoted $\pi_j \in \Pi^{\text{gen}, \det}_j$, which will give player j a significant gain in value against the policy π. To do so, we need to specify $\pi_j(h_{j,h-1}, s_h) \in \Delta(\mathcal{A}_j)$, for all $h_{j,h-1} \in \mathcal{H}_{j,h-1}$ and $s_h \in \mathcal{S}$; note that we may restrict our attention only to histories $h_{j,h-1}$ that occur with positive probability under the transitions of G.

Fix any $h_0 \in [H]$, $h_{j,h-1} \in \mathcal{H}_{j,h-1}$, and $s_h \in \mathcal{S}$. If $h_{j,h-1}$ occurs with positive probability under the transitions of G, then for each $h \in \mathcal{O}_H$, $h < h_0 - 1$ and both $j' \in [2]$, the action played by agent j' at step h is determined by $h_{j,h}$. Namely, if the state at step $h-1$ of $h_{j,h-1}$ is $s_{(a^1,a^2)}$, then player j' played action $a^{j'}$ at step h. So, for each $h \in \mathcal{O}_H$ with $h < h_0 - 1$, we may define $(a_{1,h}, a_{2,h})$ as the action profile played at step h, which is a measurable function of h_{j,h_0-1}. With this in mind, we define $\pi_{j,h_0}(h_{j,h_0-1}, s_{h_0})$ by applying Vovk's aggregating algorithm (Proposition D.2) as follows.

1. If h_0 is even, play an arbitrary action (note that the actions at even-numbered steps have no influence on the transitions or rewards).

2. If h_0 is odd, define $q_{j,h_0} \in \Delta(\mathcal{A}_j)$, by $q_{j,h_0} := \mathbb{E}_{t \sim q_{j,h_0}} [p_{j,h}]$, where $q_{j,h_0} \in \Delta(\mathcal{T})$ is defined as follows: for $t \in [T]$,

$$q_{j,h_0}(t) \propto \exp\left(-\eta \sum_{h \in \mathcal{O}_H : h < h_0} \log\left(\frac{1}{p^{(t)}_{j,h}(a_{j,h})}\right)\right) \Big/ \sum_{t'=1}^{T} \exp\left(-\eta \sum_{h \in \mathcal{O}_H : h < h_0} \log\left(\frac{1}{p^{(t')}_{j,h}(a_{j,h})}\right)\right).$$

   Note that q_{j,h_0} is a function of h_{j,h_0-1} via the action profiles $\{(a_{1,h}, a_{2,h})\}_{h \in \mathcal{O}_H : h < h_0}$; to simplify notation, we suppress this dependence.

3. Then for any state $s_{h_0} \in \mathcal{S}$, define $\pi_{j,h_0}(h_{j,h_0-1}, s_{h_0})$ to be a best response to q_{j,h_0}, namely

$$\pi_{j,h_0}(h_{j,h_0-1}, s_{h_0}) := \operatorname*{argmax}_{a_j \in \mathcal{A}_j} \mathbb{E}_{a_{\bar{j}} \sim q_{j,h_0}} R_{j,h}(s_{h_0}, (a_1, a_2)) = \operatorname*{argmax}_{a_j \in \mathcal{A}_j} \mathbb{E}_{a_{\bar{j}} \sim q_{j,h_0}} r(M_j, q_{(a_1,a_2)}). \tag{4}$$

Note that, for odd h_0, the distribution $q_{j,h_0} \in \Delta(\mathcal{A}_j)$ defined above can be viewed as an application of Vovk's online aggregation algorithm at step $h_0 - 1$ in the following setting: the number of steps (in the notation of Proposition D.2; note that T plays a different role in the present proof) is $H_0 = H/2$, the context space is $\mathcal{O}_H$, and the outcome space is $\mathcal{A}_j$.[19] There are T experts $\pi^{(1)}, \ldots, \pi^{(T)}$ (i.e., we have $I = \{\pi^{(t)}\}_{t \in [T]}$), whose predictions on a context $h \in \mathcal{O}_H$ are defined as follows: the expert $\pi^{(t)}$ predicts

[19] Here $\bar{j}$ denotes the index of the player who is not j.

$p^{ptq}(h) := p^{ptq}_{j;h}$. Then, the distribution $q_{j;p_0}$ is obtained by updating the aggregation algorithm with the context-observation pairs $(h; a_{j;h})$, for odd values of $h \geq h_0$.

We next analyze the value of $V^{\vec{\pi};\vec{\rho}}_j$ for $j \in [2]$ to show that the deviation strategy we have defined indeed obtains significant gain. To do so, recall that this value represents the payoff for player j under the process in which we draw an index $t \in [T]$ uniformly at random, then for each step $h \in [H]$, player j plays according to π_j and player j plays according to $\rho^{pt,q}_j$. (In particular, at odd-numbered steps, player j plays according to $p_{j;h}$.) We recall that $\mathbb{E}_{\vec{\pi},\vec{\rho}}[\cdot]$ denotes the expectation under this process. We let $s_{j;h+1} \in S_{j;h+1}$ denote the random variable which is the history observed by player j in this setup, i.e., when the policy played is $\vec{\pi}$, and let $\{(a_{1;h}, a_{2;h})\}_{h \in [H_0]}$ denote the action profiles for odd rounds, which are a measurable function of each player's trajectory.

We apply Proposition D.2 with the time horizon as H_0, and with the set of experts set to $I := \{p^{1,q}, \ldots, p^{T,q}\}$ as defined above. The context sequence the sequence of increasing values of $h \in [H_0]$, and for each $h \in [H_0]$, the outcome at step $(h+1)/2$ (for which the context is h) is distributed as $a_{j;h} \sim p^{ptq}(h) = p^{pt,q}_{j;h}$ conditioned on t, which in particular satisfies the realizability assumption stated in Proposition D.2. Then, since (as remarked above), the distributions $q_{j;h}$, for $h \in [H_0]$, are exactly the predictions made by Vovk's aggregating algorithm, Proposition D.2 gives that[20]

$$\mathbb{E}_{\vec{\pi},\vec{\rho}}\left[\sum_{h \in [H_0]} D_{TV}(p_{j;h}, q_{j;h}) \mid t\right] - \mathbb{E}_{\vec{\pi},\vec{\rho}^{pt,q}}\left[\sum_{h \in [H_0]} D_{TV}(p_{j;h}, p^{ptq}(h)) \mid t\right] \leq H_0 \log T. \tag{5}$$

Recall that we have assumed for the sake of contradiction that $p^{ptq}_{1;h} \times p^{ptq}_{2;h}$ is not an ε_0-Nash equilibrium of G for each $h \in [H_0]$ and $t \in [T]$. Consider a fixed draw of the random variable $t \in [T]$ defined above. Then it holds that for $j \in [2]$ and $h \in [H_0]$, defining

$$\varepsilon_{0;j;h} := \max_{a_j \in [A_j]} \mathbb{E}_{a_j \sim p_{j;h}}\left[r(M_j, q_{a_1;a_2}) \mid s\right] - \mathbb{E}_{a_1 \sim p_{1;h}, a_2 \sim p_{2;h}}\left[r(M_j, q_{a_1;a_2}) \mid s\right], \tag{6}$$

we have $\varepsilon_{0;1;h} + \varepsilon_{0;2;h} \geq \varepsilon_0$. Consider any $j \in [2]$, $h \in [H_0]$, and a history $s_{j;h+1} \in S_{j;h+1}$ of agent j up to step $h+1$ (conditioned on t). Let us write $\varepsilon_{j;h} := D_{TV}(p_{j;h}, q_{j;h})$; note that $\varepsilon_{j;h}$ is a function of $s_{j;h+1}$, through its dependence on $q_{j;h}$. We have, by the definition of $p_{j;h}(s_{j;h+1}; s_h)$ in (4) and the definition of $\varepsilon_{j;h}$,

$$\begin{aligned}
&\mathbb{E}_{a_j \sim p_{j;h}}\left[r(M_j, q_{h;j}(p^{j;h}_{\cdot;1}; s_h, a_j)) \mid t; s_{j;h+1}\right] \geq \mathbb{E}_{a_j \sim p_{j;h}}\left[r(M_j, q_{h;j}(p^{j;h+1}_{\cdot;1}; s_h, a_j)) \mid t; s_{j;h+1}\right] - \varepsilon_{j;h} \\
&\geq \max_{a_j \in [A_j]} \mathbb{E}_{a_j \sim p_{j;h}}\left[r(M_j, q_{h;j}(p^{j;h+1}_{\cdot;1}; s_h, a_j)) \mid t; s_{j;h+1}\right] - \varepsilon_{j;h} - \varepsilon_{0;j;h} \\
&\geq \max_{a_j \in [A_j]} \mathbb{E}_{a_j \sim p^{pt,q}_{h;j}}\left[r(M_j, q_{a_j;a_j})\right] - 2\varepsilon^{ptq}_{j;h}.
\end{aligned} \tag{7}$$

Combining (6) and (7), we get that for any fixed $h \in [H_0]$, $j \in [2]$, and $s_{j;h+1} \in S_{j;h+1}$,

$$\mathbb{E}_{a_j \sim p_{j;h}}\left[r(M_j, q_{h;j}(p^{j;h+1}_{\cdot;1}; s_h, a_j)) \mid t; s_{j;h+1}\right] \geq \mathbb{E}_{a_1 \sim p_{1;h}, a_2 \sim p_{2;h}}\left[r(M_j, q_{a_1;a_2})\right] - 2\varepsilon^{pt,q}_{j;h}. \tag{8}$$

[20] In fact, Proposition D.2 implies that a similar bound holds uniformly for each possible realization of t, but (5) suffices for our purposes.

Averaging over the draw of $t \Pr T s$, which we recall is chosen uniformly, we see that

$$
\begin{aligned}
& , \quad V_j^{j} \; V_{j\, j \Pr 2 s} \quad - \\
& \frac{1}{T} \, , \, {}^{T} \quad , \quad V_j^{\;:\;} \; {}_{j}^{\;ptq} V_j^{\;ptq} \qquad (9) \\
& 1 \, , \, {}^{T}, \quad E_{\;:\;} \; {}_{j}^{\;ptq} \; {}_{j} \quad , \quad E_{a_j \; p \; {}_{j;h}^{\;ptq}} \; r R_{j;h} p s; p^{\;:\;} \; p_{j;h1}; s q; a_j q q \; | t; {}_{j;h1} s E_{\;a_1 p \; {}_{1;h}^{\;ptq}; a_2 p \; {}_{2;h}^{\;ptq}} \; r R_{j;h} p s; p a_1; a_2 q q s \\
& \frac{1}{T H} \, , \, {}^{T}, \quad E_{:ptq} \quad , \quad E^{a} \; {}_{j} \; {}_{j} \; p^{ptq}{}_{j;h} \; r p M_j q_{:} \; {}_{p_{j;h}^{\;j;h1}; s q; a_j} \; | t; {}_{j;h1} s E_{a \; p^{ptq}; a \; p^{ptq}} \; r p M_j q_{a_1; a_2 \; 1 \; 2 \; ;h} s \; j \Pr 2 s \\
& \text{¥} \; \frac{1}{T H} \, , \, {}^{T} \quad , \quad E_{:ptq} \quad , \quad 0; j; h \; 2_{j;h} \qquad (10) \\
& \text{¥} \; \frac{0}{2} \; T H \; \frac{2}{t1} \; 2^{\;T} \, , \, {}^{a} \; H_0 \log T \; \text{¥} \; {}_{2} \; 4 \; {}^{a}_{0} \log p T q \{ H; \qquad (11)
\end{aligned}
$$

where (9) follows from the definition $\frac{1}{T}{}^{\circ}{}_{t1}^{T} \; l_{ptq}$, (10) follows from (8), and (11) uses (5). As long as $T \quad \exp p H p_0 \{ 16 q^2 q$, the this expression is bounded below by $_0\{4$, meaning that is not an $_0\{4$-approximate CCE. This completes the contradiction. □

E. Proofs of lower bounds for SPARSECCE (Sections 4 and 5)

In this section we prove our computational lower bounds for solving the SPARSECCE problem with $m \; 3$ players (Theorem 4.3 and Corollary 4.4), as well as our statistical lower bound for solving the SPARSECCE problem with a general number m of players (Theorem 5.2).

Both theorems are proven as consequences of a more general result given in Theorem E.1 below, which reduces the NASH problem in m-player normal-form games to the SPARSECCE problem in $pm \quad 1q$-player Markov games. In more detail, the theorem shows that (a) if an algorithm for SPARSECCE makes few calls to a generative model oracle, then we get an algorithm for the NASH problem with few calls to a payoff oracle (see Section C.3 for background on the payoff oracle for the NASH problem), and (b) if the algorithm for SPARSECCE is computationally efficient, then so is the algorithm for the NASH problem.

Theorem E.1. There is a constant $C_0 \; 0$ so that the following holds. Consider $n; m \; P \; N$, and suppose $T; N; Q \; P \; N$ and $\; 0$ satisfy $1 \quad T \quad \exp \frac{{}^{2} tn\{mu}{m^2}$. Suppose there is an algorithm B which, given a generative model oracle for a $pm \quad 1q$-player Markov game G with $|G| / n$, solves the $pT; ; N q$-SPARSECCE problem for G using Q generative model oracle queries. Then the following conclusions hold:

- For any $\; 0$, the m-player $ptn\{mu; 16pm \quad 1qq$-NASH problem for any normal-form game G can be solved, with failure probability , using at most $C_0 pQlogp1\{qq \quad plogp1\{qnm\{q^{C_0}$ queries to a payoff oracle O_G for G.
- If the algorithm B additionally runs in time U for some $U \; P \; N$, then the algorithm solving NASH from the previous bullet point runs in time $pnmTNUlogp1\{q\{q^{C_0}$.

Theorem 4.3 follows directly from Theorem E.1 by taking $m \; 2$.

Proof of Theorem 4.3. Suppose there is an algorithm which, given the description of any 3-player Markov game G with $|G| / n$, solves the $pT; ; N q$-SPARSECCE problem in time U. Such an algorithm immediately yields an algorithm which can solve the $pT; ; N q$-SPARSECCE problem in time $U \quad |G|^{Op1q}$ using only a generative model oracle, since the exact description of the Markov game can be obtained with $H S |A| / H S p max_i A_i q^3 / |G|^5$ queries to the generative model (across all $ph; s; aq$ tuples). We can now solve the problem of computing a 50-Nash equilibrium of a given 2-player $tn\{2u$-action normal form game G as follows. We simply apply the algorithm of Theorem E.1 with $m \; 2$, noting that the oracle O_G in the theorem statement can be implemented by reading the corresponding bits of input the input game G. The second bullet point yields that this algorithm

takes time $pnT\,NU\log p1\{q\{q^{C_0}$, for some constant C_0. Furthermore, the assumption $T \quad \exp p^2 tn\{mu\{m^2 q$ of Theorem E.1 is implied by the assumption that $T \quad \exp p^2 n\{16q$ of Theorem 4.3. □

In a similar manner, Theorem 5.2 follows from Theorem E.1 by applying Theorem C.3, which states that there is no randomized algorithm that finds approximate Nash equilibria of m-player, 2-action normal form games in time 2^{opmq}.

Proof of Theorem 5.2. Let $_0$ be the constant from Theorem C.3, and consider any $m \yen 3$. Suppose there is an algorithm which, for any m-player Markov game G with $|G| \diagup 2m^6$, makes Q oracle queries to a generative model oracle for G, and solves the $pT;_0\{p10mq;Nq$-SPARSECCE problem for G for some $T;N \; PN$ so that $T \quad \exp ppcmq$, for a sufficiently small absolute constant c. Then, by Theorem E.1 with $_0\{p10mq$ and $n \; m^6$ (which ensures that $T \quad \exp pp_0\{p10mqq^2 \; tn\{mu\{m^2 q$ as long as c is sufficiently small), there is an algorithm which solves the $pm^5;_0q$-NASH problem—and thus the $p2;_0q$-NASH problem—for $pm\,1q$-player games with failure probability $1\{3$, using $OpQq \quad m^{Op1q}$ queries to a payoff oracle. But by Theorem C.3, any such algorithm requires 2^{pmq} queries to a payoff oracle. It follows that $Q \yen 2^{pmq}$, as desired. □

E.1. Proof of Theorem E.1

Proof of Theorem E.1. Fix any $m \yen 2$, $n \; PN$. Suppose we are given an algorithm B that solves the $pm \quad 1q$-player $pT;;N\,q$-SPARSECCE problem for Markov games G satisfying $|G| \diagup \quad n$, running in time U and using at most Q generative model queries. We proceed to describe an algorithm which solves the m-player $ptn\{mu;16pm \quad 1q\,q$-NASH problem using $C_0 \; pQ\log p1\{qq \quad plogp1\{q\,nm\{q^{C_0}$ queries to a payoff oracle, and running in time $pnmT\,NU\log p1\{q\{q^{C_0}$, where represents the failure probability. Define $n_0 :tn\{mu$, and assume we are given an arbitrary m-player n_0-action normal form G, which is specified by payoff matrices $M_1;:::;M_m \; Pr0;1s^{n_0 n_0}$. We assume that all entries of each of the matrices M_j have only the most significant $maxtn_0;rlog1\{su$ bits nonzero; this assumption is without loss of generality, since by truncating the utilities to satisfy this assumption, we change all payoffs by at most , which degrades the quality of any approximate equilibrium by at most 2 (in addition, we have $rlog1\{s/n_0$ since we have assumed $1 \quad T \quad \exp p^2 n_0\{m^2 q$). We assume $\diagup 1\{2$ without loss of generality. Based on G, we construct an $pm \quad 1q$-player Markov game $G :GpGq$ as follows.

Definition E.2. We define the Markov game GpGq as the tuple $GpGq\,pS;H;pA_i q_{iPr2s};P;pR_i q_{iPr2s};q$, where:

- The horizon of G is chosen to be the power of 2 satisfying $n_0 \diagup H \quad 2n_0$.
- Let $A :n_0$. The action spaces of agents $1;2;:::;m$ are given by $A_1 \; A_m \; rAs$. The action space of agent $m \quad 1$ is

$$A_{m \quad 1} tpj;a_j q \; : \; j \; Prms;a_j \; PA_j u;$$

  so that $|A_{m \quad 1}| \, Am \diagup n$.

  We write $A \quad \overline{}^{m}_{j1} \; A_j$ to denote the joint action space of the first m agents, and $\overline{A} : \quad \overline{}^{m \quad 1}_{j1} A_j$ to denote the joint action space of all agents.
- There is a single state, denoted by s, i.e., $S \; tsu$ (in particular, is a singleton distribution supported on s).
- For all $h \; PrHs$, the reward for agent $j \; Prm \quad 1s$, given an action profile $a\,pa_1;:::;a_{m \quad 1}q$ at the unique state s, is as follows: writing $a_{m \quad 1}\,pj^1;a^1{}_1 q_j$, we have

$$R_{j;h}ps;aq \; \overline{R}_{j;h}ps;aq \quad H^{\frac{1}{2}3rlog1\{s} encpaq; \tag{12}$$

  where $R_{j;h}ps;aq$ is defined per the kibitzer construction of (Borgs et al., 2008):

$$\overline{R}_{j;h}ps;aq : \begin{cases} 0 & : j \; Rtj^1;m \quad 1u \\ \frac{1}{H} \, pM_j q_{a_1;:::;a_m} \, pM_j q_{a^1;:::;a^1{}_1;:::;a_m} & : j \; j^1 \\ \frac{1}{H} \quad pM_j q_{a_1;:::;a^1{}_j;:::;a_m} \, pM_j q_{a_1;:::;a_m} & : j \; m \quad 1: \end{cases} \tag{13}$$

  In (12) above, $encpaq \; Pr0;1s$ is the binary representation of a binary encoding of the action profile a. In particular, if the binary encoding of a is $pb_1;:::;b_N q$, with $b_i \; Pt0;1u$, then $encpaq \quad \sum_{i1}^{N} 2^i b_i$. Note that encpaq takes $N \; Opmlogn_0 q \diagup Opmlognq$ bits to specify.

Algorithm 2 Algorithm to compute Nash equilibrium used in proof of Theorem E.1.

Input:

Parameters $n, n_0, m, T \in \mathbb{N}$, $\{p6Hq$, $K \; r4\log(mn_0\{q\{^2s$.

An m-player, n_0-action normal form game G, with utilities accessible by oracle O_G.

An algorithm B for computing approximate CCE of Markov games.

Call the algorithm B on the $(m+1)$-player Markov game $G' = \text{GpGq}$ constructed as in Definition E.2, which produces a sequence $\pi^{1q}, \ldots, \pi^{Tq}$, where each $\pi^{tq} = (\pi^{tq}, \ldots, \pi^{tq}_{m+1q})$ with $\pi^{tq}_m \in \Pi^{gen;rnd}_j$. Here, we use the oracle O_G to simulate generative model oracle queries made by B.

Draw $t \in [T]$ uniformly at random.

For each $j \in [m]$, initialize $\tau_{j;0}$ to be an empty trajectory.

for $h \in [H]$: do

Set $s_h = s$ (per the transitions of G).

For each $j \in [m]$, define $\mu_{j;h} := \mathbb{E}_{\tau \sim q_{j;h}} \pi^{tq}_{j;h}(\tau, s_h) \in \Delta(\mathcal{A}_j)$, where $q_{j;h} \in \Delta(\mathcal{T}_{j;h})$ is defined as follows: for $\tau \in \mathcal{T}_j$,

$$q_{j;h}(\tau) := \frac{\exp\left(\eta \sum_{g=h}^{H} \log \frac{1}{\pi^{tq}_{j;g}(a_{j;g} \mid \tau_{j;g-1}, s_g)}\right)}{\sum_{t=1}^{T} \exp\left(\eta \sum_{g=h}^{H} \log \frac{1}{\pi^{tq}_{j;g}(a_{j;g} \mid \tau_{j;g-1}, s_g)}\right)}$$

Draw K i.i.d. samples $a^1_h, \ldots, a^K_h \sim \prod_{j=1}^{m} q_{j;h}$.

For each $a^1 \in \mathcal{A}_{m+1}$, define $\widehat{R}_{m+1;h}(a^1) := \frac{1}{K} \sum_{k=1}^{K} R_{m+1;h}(s_h, (a^k_h, a^1))$. Here, we use the oracle O_G to compute $R_{m+1;h}(s_h, (a^k_h, a^1))$ for each tuple (a^k_h, a^1).

For each $j \in [m]$, draw $a_{j;h} \sim \pi^{tq}_{j;h}(\tau_{j;h-1}, s_h)$.

Choose the action $a_{m+1;h}$ of player $m+1$ as follows: (Action $a_{m+1;h}$ is corresponds to the action selected by the policy π^{tq}_{m+1} of player $m+1$ defined within the proof of Lemma E.3; this policy is well-defined because the action profiles of all players $i \in [m]$ can be extracted from the lower-order bits of player $m+1$'s reward)

$$a_{m+1;h} := \operatorname*{argmax}_{a^1 \in \mathcal{A}_{m+1}} \left(\widehat{R}_{m+1;h}(a^1) \right) \tag{14}$$

For each $j \in [m+1]$, let $r_{j;h} = R_{j;h}(s_h, (a_{1;h}, \ldots, a_{m+1;h}))$.

Each player j constructs $\tau_{j;h}$ by updating $\tau_{j;h-1}$ with $(s_h, a_{j;h}, r_{j;h})$.

if $\widehat{R}_{m+1;h}(a_{m+1;h}) / (16(m+1)) \geq \{H$ then

return $\mu_h := \prod_{j \in [m]} \mu_{j;h}$ as a candidate approximate Nash equilibrium for G.

end if

end for

if the for loop terminates without returning: return fail.

It is evident that this construction takes polynomial time and satisfies $|G'| / (mn_0 / n)$. Furthermore, it is clear that a single generative model oracle call for the Markov game G' (per Definition 5.1) can be implemented using at most 2 calls to the oracle O_G for the normal-form game G. We will now show by applying the algorithm B to G', we can efficiently (in terms of runtime and oracle calls) compute a $(16(m+1)-1q)$-approximate Nash equilibrium for the original game G. To do so, we appeal to Algorithm 2.

Algorithm 2 proceeds as follows. First, it calls the algorithm B on the $(m+1)$-player Markov game $G' = \text{GpGq}$, using the oracle O_G to simulate B's calls to the generative model oracle for G. By assumption, the algorithm B returns a sequence $\pi^{1q}, \ldots, \pi^{Tq}$ of product policies of the form $\pi^{tq} = (\pi^{tq}, \ldots, \pi^{tq}_{m+1q})$, so that each $\pi^{tq}_j \in \Pi^{gen;rnd}_j$ is N_j-computable, and so that the average $\frac{1}{T} \sum_{t=1}^{T} \ldots$

is an ε-CCE of G. Next, Algorithm 2 samples a trajectory from G in which:

- Players $1,\dots,m$ each play according to a policy π^{t} for an index $t \in [T]$ chosen uniformly at the start of the episode.
- Player $m+1$ plays according to a strategy that, at each step $h \in [H]$, computes distributions $q_{j,h}$ representing its "belief" of what action each player $j \in [m]$ will play at step h (Line 10), and plays an approximate best response to the product of the strategies $q_{j,h}$, $j \in [m]$ (Line 14).

In order avoid exponential dependence on the number of players m, when computing an approximate best response to $\prod_{j\in[m]} q_{j,h}$ we draw $K = 4\log(mn_0)/\varepsilon^2$ samples from $\prod_{j\in[m]} q_{j,h}$ and use these samples to compute the best response. In particular, letting $a_h^k \in A$ denote the kth sampled action profile, we construct a function $R_{m+1,h} : A_{m+1} \to \mathbb{R}$ in Lines 11 and 14 which, for each $a^{m+1} \in A_{m+1}$, is defined as the average over samples $\{a^k\}_{k\in[K]}$ of the realized payoffs $R_{m+1,h}(s_h, (a^k, a^{m+1}))$; note that to compute the payoffs for each sample, Algorithm 2 needs only two oracle calls to O_G.

The following lemma, proven in the sequel, gives a correctness guarantee for Algorithm 2.

Lemma E.3 (Correctness of Algorithm 2). *Given any m-player n_0-action normal form game G, if the algorithm B solves the (T, N, ε)-SPARSECCE problem for the game $G(G)$ with T, N satisfying $T/\exp(n_0^2)\ge (m^2/\varepsilon)$, then Algorithm 2 outputs a $16(m+1)\varepsilon$-approximate Nash equilibrium of G with probability at least $1/3$, and otherwise fails.*

The assumption that $T \ge \exp^{2}(n_0^{m})$ from the statement of Theorem E.1 yields that $T/\exp(n_0^2)\ge (m^2/\varepsilon)$, so Lemma E.3 yields that Algorithm 2 outputs a $16(m+1)\sqrt{\varepsilon}$-Nash equilibrium of G with probability at least $1/3$ (and otherwise fails). By iterating Algorithm 2 for $\log(1/\delta)$ times, we may thus compute a $16(m+1)\sqrt{\varepsilon}$-Nash equilibrium of G with failure probability δ.

We now analyze the oracle cost and computational cost of Algorithm 2. It takes $2Q$ oracle calls to O_G to simulate the Q generative model oracle calls of B, and therefore, if B runs in time U, then the call to B on Line 5, using oracle calls to O_G to simulate simulate the generative model oracle calls, runs in time $O(U)$. Next, the computations of $q_{j,h}$ (and thus $q_{j,h}(\pi)$) in Line 10 can be performed in $(nmTN)^{O(1)}$ time, the computation of $R_{m+1,h} : A_{m+1} \to \mathbb{R}$ in Line 14 requires time (and oracle calls to O_G) bounded above by $O(|A_{m+1}|K) = O(nm\log(1/\delta)/\varepsilon^2)$, constructing the actions $a_{j,h}$ (for $j \in [m+1]$) in Lines 13 and 14 takes time $(Nmn)^{O(1)}$ (using the fact that the policies $\pi_{j,h}$ are N-computable), and constructing the rewards $r_{j,h}$ on Line 15 requires another $2(m+1)$ oracle calls to O_G. Altogether, Algorithm 2 requires $2Q + (nm\log(1/\delta)/\varepsilon^2)$ oracle calls to O_G and, if B runs in time U, then Algorithm 2 takes time $(nmTNU\log(1/\delta)/\varepsilon)^{C_0}$, for some absolute constant C_0.

Remark E.4 (Bit complexity of exponential weights updates). In the above proof we have noted that $q_{j,h}$ (as defined in Line 10 of Algorithm 2) can be computed in time $(nmTN)^{O(1)}$. A detail we do not handle formally is that, since the values of $q_{j,h}(\pi)$ are in general irrational, only the $(nmTN)^{O(1)}$ most significant bits of each real number $q_{j,h}(\pi)$ can be computed in time $(nmTN)^{O(1)}$. To give a truly polynomial-time implementation of Algorithm 2, one can compute only the $(nmTN)^{O(1)}$ most significant bits of each distribution $q_{j,h}$, which is sufficient to approximate the true value of $q_{j,h}(\pi)$ to within $\exp(-(nmTN)^{O(1)})$ in total variation distance. Since $q_{j,h}$ only influences the subsequent execution of Algorithm 2 via the samples $a^1_h,\dots,a^K_h \sim \prod_{j\in[m]} q_{j,h}$ drawn in Line 11, by a union bound, the approximation of $q_{j,h}$ we have described perturbs the execution of the algorithm by at most $O(KH)\exp(-(nmTN)^{O(1)})$ in total variation distance. In particular, the correctness guarantee of Lemma E.3 still holds, with sucess probability at least $1/3-\exp(-(nmTN)^{O(1)}) \ge 1/4$.

It remains to prove Lemma E.3, which is the bulk of the proof of Theorem E.1.

Proof of Lemma E.3. We will establish the following two facts:

1. First, the choices of $a_{m+1,h}$ in Line 14 (i.e., Eq. 14) of Algorithm 2 correspond to a valid policy $\pi_{m+1} \in \Pi^{\mathrm{gen,rnd}}$ for player $m+1$ (representing a strategy for deviating from the equilibrium π), in that they can be expressed as a function of player $(m+1)$'s history, $(s_{m+1,1}, a_{m+1,1}, \dots, s_h)$ at each step h.
2. Second, we will show that, since π is an ε-CCE of G, the strategy π_{m+1} cannot not lead to a large increase of value for player $m+1$, which will imply that Algorithm 2 must return a Nash equilibrium with high enough probability.

Defining π_i for $i \in [m-1]$. We begin by constructing the policy π_{m-1} described; for later use in the proof, it will be convenient to construct a collection of closely related policies $\pi_i^{\mathrm{gen,rnd}}$ for $i \in [m]$, also representing strategies for deviating from the equilibrium π.

Let $i \in [m-1]$ be fixed. For $h \in [H]$, the mapping $\pi_{i,h} : \mathcal{H}_{i,h-1} \times \mathcal{S} \to \mathcal{A}_i$ is defined as follows. Given a history $\tau_{i,h-1} = (s_1, a_{i,1}, r_{i,1}, \ldots, s_{h-1}, a_{i,h-1}, r_{i,h-1}) \in \mathcal{H}_{i,h-1}$ (we assume without loss of generality that $\tau_{i,h-1}$ occurs with positive probability under some sequence of general policies) and a current state s_h, we define $\pi_{i,h}(\tau_{i,h-1}, s_h) \in \mathcal{A}_i$ through the following process.

1. First, we claim that for all players $j \in [m]$, it is possible to extract the trajectory $\tau_{j,h-1}$ from the trajectory $\tau_{i,h-1}$ of player i.
   (a) Recall that for each $g \le h$, from the definition in (12) and the function $\mathrm{enc}(a_g)$, the bits following position $3r\log(1/\delta)$ of the reward $r_{i,g}$ given to player i at step g of the trajectory $\tau_{i,g-1}$ encode an action profile $a_g \in \mathcal{A}$. Since $\tau_{i,h-1}$ occurs with positive probability, this is precisely the action profile which was played by agents at step g. Note we also use here that by definition of the rewards $R_{j,h}(s,a)$ in (12), the component $R_{j,h}(s,a)$ of the reward only affects the first $2r\log(1/\delta)$ bits.
   (b) For $g \le h$ and $j \in [m]$, define $r_{j,g} := R_{j,g}(s_g, a_g)$.
   (c) For $j \in [m]$, write $\tau_{j,h} := (s_1, a_{j,1}, r_{j,1}, \ldots, s_{h-1}, a_{j,h-1}, r_{j,h-1})$; in particular, $\tau_{j,h-1}$ is a deterministic function of $(\tau_{i,h-1}, s_h)$. (Note that, since $\tau_{i,h-1}$ occurs with positive probability, the history $\tau_{j,h-1}$ observed by player j up to step $h-1$ can be computed from it via Steps (a) and (b)). Going forward, for $g \le h-1$, we let $\tau_{j,g}$ denote the prefix of $\tau_{j,h-1}$ up to step g.
2. Now, using that player i can compute all players' trajectories, for each $j \in [m]$ we define
$$\widehat{p}_{j,h} := \mathbb{E}_{t \sim q_{j,h}} \widehat{p}_{j,h}(t, s_h) \in \Delta(\mathcal{A}_j), \tag{15}$$
where $q_{j,h} \in \Delta(\mathcal{T})$ is defined as follows: for $t \in \mathcal{T}$,
$$q_{j,h}(t) := \frac{\exp\left(\gamma \sum_{g=1}^{h} \log \frac{1}{p(a_{j,g} \mid \tau_{j,g-1}, s_g)}\right)}{\sum_{t' \in \mathcal{T}} \exp\left(\gamma \sum_{g=1}^{h} \log \frac{1}{p(a'_{j,g} \mid \tau'_{j,g-1}, s'_g)}\right)} \tag{16}$$
Note that $\widehat{p}_{j,h}$ is a random variable which depends on the trajectory $(\tau_{j,h-1}, s_h)$ (which can be computed from $(\tau_{i,h-1}, s_h)$). In addition, the definition of $\widehat{p}_{j,h}$ (for each $j \in [m]$) is exactly as is defined in Line 10 of Algorithm 2.
3. For $i \in [m]$, define $\pi_{i,h}(\tau_{i,h-1}, s_h)$ as follows:
$$\pi_{i,h}(\tau_{i,h-1}, s_h) := \operatorname*{argmax}_{a^1 \in \mathcal{A}_i} \mathbb{E}_{a_i \sim \pi_i(\tau_{i,h-1}, s_h),\, \{a_j \sim \widehat{p}_{j,h}\}_{j \neq i}} R_{m-1,h}(s_h, (a^1, a_i, a_{-i,-1})). \tag{17}$$
For the case $i = m-1$, define $\pi_{m-1,h}(\tau_{m-1,h-1}, s_h) \in \Delta(\mathcal{A}_{m-1})$ (implicitly) to be the following distribution over $a_{m-1,h} \in \mathcal{A}_{m-1}$: draw $a^1, \ldots, a^K \sim \bigotimes_{j \in [m]} \widehat{p}_{j,h}$, define $\widehat{R}_{m-1,h}(a^1) := \frac{1}{K} \sum_{k=1}^{K} R_{m-1,h}(s_h, (a^k, a^1_{-m}))$ for $a^1 \in \mathcal{A}_{m-1}$, and finally set
$$a_{m-1,h} := \operatorname*{argmax}_{a^1 \in \mathcal{A}_{m-1}} \widehat{R}_{m-1,h}(a^1). \tag{18}$$
Note that, for each choice of $(\tau_{m-1,h-1}, s_h)$, the distribution $\pi_{m-1,h}(\tau_{m-1,h-1}, s_h)$ as defined above coincides with the distribution of the action $a_{m-1,h}$ defined in Eq. 14 in Algorithm 2, when player $m-1$'s history is $\tau_{m-1,h-1}$ and the state at step h is s_h. The following lemma, for use later in the proof, bounds the approximation error incurred in sampling $a^1, \ldots, a^K \sim \bigotimes_{j \in [m]} \widehat{p}_{j,h}$.

Lemma E.5. Fix any $(\tau_{m-1,h-1}, s_h) \in \mathcal{H}_{j,h-1}$. With probability at least $1-\delta$ over the draw of $a^1, \ldots, a^K \sim \bigotimes_{j \in [m]} \widehat{p}_{j,h}$, it holds that for all $a^1 \in \mathcal{A}_{m-1}$,
$$\left|\widehat{R}_{m-1,h}(a^1) - \mathbb{E}_{a_j \sim \widehat{p}_{j,h}} R_{m-1,h}(s_h, (a_1, \ldots, a_m, a^1))\right| \le \sqrt{\frac{\log(|\mathcal{S}|/\delta)}{K}}.$$

which implies in particular that with probability at least 1 over the draw of $a_{m:1;h}\, m_{1;h}\, p_{m:1;h1}; s_h q$,

$$\max_{a^1 P A_{m:1}} E_{a_j \varphi_{j;h} @ j P r m s r R_{m:1;h} p s_h; p a_1; :::; a_m; a^1 q q s} \left({}^{2} / E_{\overline{H}} \right)_{a_j \varphi_{j;h} @ j P r m s r R_{m:1;h} p s_h; p a_1; :::; a_m; a_{m:1;h} q q s}. \tag{19}$$

It is immediate from our construction above that the following fact holds.

Lemma E.6. The joint distribution of $_{j;h}$, for $j P r m:1s$ and $h P r H s$, as computed by Algorithm 2, coincides with the distribution of $_{j;h}$ in an episode of G when players follow the policy $_{m:1} p m:1q$.

Analyzing the distributions $p_{j;h}$. Fix any $i P r m:1s$. We next prove some facts about the distributions $q_{j;h}$ defined above (as a function of $p_{i;h1}; s_h q$) in the process of computing $_{i;h} p_{i;h1}; s_h q$.

For each $h P r H s$, consider any choice of $p_{i;h1}; s_h q P H_{i;h1} S$; note that for each $j P r m:1s$, the distributions $q_{j;h} P p A_j q$ for $h P r H s$ may be viewed as an application Vovk's aggregating algorithm (Proposition D.2) in the following setting: the number of steps (T, „ in the context of Proposition D.2; note that T has a different meaning in the present proof) horizon is H, the context space is $_{h1}^{H} H_{j;h1} S$, and the output space is A_j. The expert set is $I t^{p1q}; :::; {}^{pTq} u_j$ (which has $|I| T$), and the experts' predictions on a context $p_{j;h1}; sq P H_{j;h1} S$ are defined via $_j {}^{ptq} p|_{j;h1}; sq := {}^{ptq} p|_{j;h1}; sq P p A_j q$. Then for each $h P r H s$, the distribution $p_{j;h}$ is obtained by updating the aggregating algorithm with the context-observation pairs $p_{j;h1}; a_{j;h1} q$ for $h^1 1; 2; :::; h1$.

In more detail, fix any $t P r T s$ and $j P r m:1s$ with $i \neq j$. We may apply Proposition D.2 with the number of steps set to H, the set of experts as $I_j t^{p1q}; :::; {}_j^{pTq} u$, and contexts and outcomes generated according to the distribution induced by running the policy $_{i\, i}^{:q}$ in the Markov game G as follows:

- For each $h P r H s$, we are given, at steps $h^1 \neq h$, the actions $a_{k;h1}$ rewards $r_{k;h1}$ for all agents $k P r m:1s$, as well as the states $s_1; :::; s_h$.
  - For each $k P r m:1s$, set $_{k;h1} p s_1; a_{k;1}; r_{k;1}; :::; s_{h1}; a_{k;h1}; r_{k;h1} q$ to be agent k's history. – The context fed to the aggregation algorithm at step h is $p_{j;h1}; s_h q$.
  - The outcome at step h is given by $a_{j;h} {}^{ptq} p|_{j;h1}; s_h q$; note that this choice satisfies the realizability assumption in Proposition D.2.
  - To aid in generating the next context at step $h \neq 1$, choose $a_{k;h} {}_{k;h}^{t} p_{k;h1}; s_h q$ for all $k P r m:1szti; ju$ and $a_{i;h} {}_{i;h}^{:} p_{i;h1}; s_h q$. Then set $s_{h \neq 1}$ to be the next state given the transitions of G and the action profile $a_h p a_{1;h}; :::; a_{m:1;h} q$.

By Proposition D.2, it follows that for any fixed $t P r T s$ and $j P r m:1s$ with $j \neq i$, under the process described above we have

$$E_{:ptq\ i\ i} \sum_{h1}^{H} D_{TV} p^{ptq} p_{j;h1}; s_h q; p_{j;h} q \;/\; \underline{a\ H \log T:} \tag{20}$$

Analyzing the value of $_{m:1}$. Next, using the development above, we show that if Algorithm 2 successfully computes a Nash equilibrium with constant probability (via $_{m:1}$) whenever s is an ε-CCE. We first state the following claim, which is proven in the sequel by analyzing the values $V_{i\ i}$ for $i P r m s$.

Lemma E.7. If $\bar{\pi}$ is an ε-CCE of G, then it holds that for all $i P r m s$,

$$V_i^{\bar{\pi}} \leq m \sqrt{\log p T q / H:}$$

Note that in the game G, since for all $h P r H s$, $s P S$ and $\bar{a} P A$, it holds that $\sum_{j1}^{m} R_{j;h} p s; a q \leq \frac{pm \neq 1q^2}{H}$ (which holds since in (12), $\varepsilon_{\text{enc}} p a q$ is multiplied by $_H 2^{3r \log 1\{s}$), it follows that $\sum_{j1} V \leq pm \neq 1q^2$. Thus, by Lemma E.7, we have $V_{m:1}^{\bar{\pi}} \leq pm \neq 1q^2 \leq m \sqrt{\log p T q / H q}$, and since $\bar{\pi}$ is an ε-CCE of G it follows that

$$V_{m:1}^{m:1\, pm\ 1q} \leq 2pm \neq 1q \leq m^2 \sqrt{\log p T q / H:} \tag{21}$$

To simplify notation, we will write $q_h := q_{1;h-1}$ in the below calculations, where we recall that each $q_{j;h}$ is deterministic given the history up to step h, $p_{j;h+1}(s_h)$, as defined in (15) and (16). An action profile drawn from q_h is denoted as $a \sim q_h$, with $a \in \mathcal{A}$. We may now write $V_1^{\pi_{1:m},\mu_{1:m}}$ as follows:

$$
\begin{aligned}
V^{\pi_{1:m},\mu_{1:m}} &= \frac{1}{H}\, \mathbb{E}_{\tau \sim \mathbb{P}^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}} \sum_{h=1}^{H} r_{m+1}(s_h, a_{1:m+1,h}) \\
&= \frac{1}{H}\, \mathbb{E}_{\tau \sim \mathbb{P}^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}} \sum_{h=1}^{H} \mathbb{E}_{a_{j,h} \sim \pi_{j,h}(\cdot \mid p_{j;h+1}(s_h), q_{j,h})} r_{m+1}(s_h, a_{1:m+1,h}) \\
&= \frac{1}{H}\, \mathbb{E}_{\tau \sim \mathbb{P}^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}} \sum_{h=1}^{H} \mathbb{E}_{a_{m+1,h} \sim \mu_{m+1,h}(p_{m+1,1:H}(s_h), q_{m+1,h})} r_{m+1}(s_h, a_{1:m+1,h}) \\
&\quad \pm \frac{1}{H} \sum_{j \in [m]} D_{\mathrm{TV}}\big(p^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}(p_{j;h+1}(s_h), q_{j,h}, q_{-j,h}) \,\big\|\, p^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}(p_{j;h+1}(s_h), q_{j,h}, q'_{-j,h})\big) \\
&\leq \frac{1}{H}\, \mathbb{E}_{q'_{-1} \sim q_{-1}} \sum_{h=1}^{H} \max_{a^1 \in \mathcal{A}} \mathbb{E}_{a \sim \pi_h} R_{m+1,h}(s_h, [a, a^1_{-1,h}], q'_h) \pm \frac{m}{H} \\
&\quad \pm \frac{1}{H} \sum_{j \in [m]} D_{\mathrm{TV}}\big(p^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}(p_{j;h+1}(s_h), q_{j,h}, q_{-j,h}) \,\big\|\, p^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}(p_{j;h+1}(s_h), q_{j,h}, q'_{-j,h})\big) \\
&\leq \frac{1}{H}\, \mathbb{E}_{\tau \sim \mathbb{P}^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}} \sum_{h=1}^{H} \max_{a^1_j \in \mathcal{A}_j,\, j \in [m]} \mathbb{E}_{a \sim \pi_h} r_{m+1}(M_j(q_{a^1_j, a_{-j}}), q_j, s_a) \pm \frac{m}{H} \pm \sqrt{H \log T \cdot 2^2},
\end{aligned}
$$

where:

- The first inequality follows from the fact that $R_{m+1,h}(\cdot)$ takes values in $[-1/H, 1/H]$ and the fact that the total variation between product distributions is bounded above by the sum of total variation distances between each of the pairs of component distributions.
- The second inequality follows from the inequality (19) of Lemma E.5.
- The final equality follows from the definition of the rewards in (12) and (13), and by summing (20) over $j \in [m]$. We remark that the 2^2 term in the final line comes from the term $\frac{2}{H} \cdot 2\sqrt{T \log(1/\delta)}$ encapsulated in (12).

Rearranging and using (21) as well as the fact that $2/(6H) \leq 2/(3H)$, we get that

$$
\begin{aligned}
\mathbb{E}_{\tau \sim \mathbb{P}^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}} \sum_{h=1}^{H} \max_{a^1_j \in \mathcal{A}_j,\, j \in [m]} \mathbb{E}_{a \sim \pi_h} r_{m+1}(M_j(q_{a^1_j, a_{-j}}), q_j, s_a) \\
\leq 2Hm(1+m) + \sqrt{H \log T} \leq 3H.
\end{aligned}
$$

Since π_h is a product distribution a.s., we have that

$$
\max_{a^1_j \in \mathcal{A}_j,\, j \in [m]} \mathbb{E}_{a \sim \pi_h} r_{m+1}(M_j(q_{a^1_j, a_{-j}}), q_j, s_a) \geq 0.
$$

Therefore, by Markov's inequality, with probability at least $1/2$ over the choice of $\tau \sim \mathbb{P}^{\mathrm{ptq}}_{\pi_{1:m},\mu_{1:m}}$ and the trajectories $p_{j;h+1}(s_h), q_{m+1:m+1,1:q}$ for $j \in [m]$ (which collectively determine q_h), there is some $h \in [H]$ so that

$$
\max_{a^1_j \in \mathcal{A}_j,\, j \in [m]} \mathbb{E}_{a \sim \pi_h} r_{m+1}(M_j(q_{a^1_j, a_{-j}}), q_j, s_a) \leq 10m(1+2m(1+m)) \sqrt{\log(T)/(H/12m(1+m))}; \tag{22}
$$

where the final inequality follows as long as $H \geq m^2 \log T$, i.e., $T \geq \exp(H/m^2)$, which holds since $H \geq n_0$ and we have assumed that $T \geq \exp(n_0/m^2)$.

Note that (22) implies that with probability at least 1{2 under an episode drawn from m 1pm 1q, there is some h P rHs so that q_h is a 12pm 1q-Nash equilibrium of the stage game G. Thus, by Lemma E.6, with probability at least 1{2 under an episode drawn from the distribution of Algorithm 2, there is some h P rHs so that p_h is a 12pm 1q-Nash equilibrium of G.

Finally, the following two observations conclude the proof of Lemma E.3.

- If p_h is a 12pm 1q-Nash equilibrium of G, then by definition of the reward function $R_{m\ 1;h}$pq in (12), upper bounding H 13rlog1{s encpaq by ^{2}{H,

$$\max_{a^1 P A_{m\ 1}} E_{a \sim p_h} R_{m\ 1;h}ps;pa;a^1qq \not H \quad 12pm \quad 1q \quad H ; \quad 2$$

  which implies, by Lemma E.5, that with probability at least 1 over the draw of a_h^1;:::;a_h^K,

$$\max_{a^1 P A_{m\ 1}} \hat{R}_{m\ 1;h}pa^1q \not \frac{1}{H} 12pm \quad 1q \quad \frac{2}{H} \quad H \not H \quad \frac{1}{H} 14pm \quad 1q;$$

  i.e., the check in Line 17 of Algorithm 2 will pass and the algorithm will return p_h (if step h is reached).

- Conversely, if $\max_{a^1 P A_{m\ 1}} \hat{R}_{m\ 1;h}pa^1q \not 14pm \quad 1q$, i.e., the check in Line 17 passes, then by Lemma E.5, with probability at least 1 over a_h^1;:::;a_h^K,

$$\max_{a^1 P A_{m\ 1}} E_{a \sim p_h} R_{m\ 1;h}ps;pa;a^1qq \not H \quad 14pm \quad 1q \quad H \not H \quad 15pm \quad 1q;$$

  which implies, by the definition of $R_{m\ 1;h}$pq in (12) and (13), that p_h is a 16pm 1q-Nash equilibrium of G.

Taking a union bound over all H of the probability- failure events from Lemma E.5 for the sampling a_h^1;:::;a_h^K p_h (for h P rHs), as well as over the probability-1{2 event that there is no q_h which is a 12pm 1q-Nash equilibrium of G, we obtain that with probability at least 1 1{2 H{p6Hq ¥ 1{3, Algorithm 2 outputs a 16pm 1q-Nash equilibrium of G. □

Finally, we prove the remaining claims stated without proof above.

Proof of Lemma E.5. Since $R_{m\ 1;h}$ps;aq P r1{H;1{Hs for each a P A, by Hoeffding's inequality, for any fixed $a^1 P A_{m\ 1}$, with probability at least 1{|$A_{m\ 1}$|1{pmn$_0$q over the draw of a_h^1;:::;a_h^K jPrms $p_{j;h}$, it holds that

$$\hat{R}_{m\ 1;h}pa^1q E_{a_j \sim p_{j;h}\ @j P rms} rR_{m\ 1;h}ps_h;pa_1;:::;a_m;a^1qqs \not H \quad \frac{c}{} \frac{\log 2mn_0 {}}{K} \not ; \quad H$$

where the final inequality follows from the choice of K r4logpmn$_0${q{^{2}s. The statement of the lemma follows by a union bound over all |$A_{m\ 1}$| actions $a^1 P A_{m\ 1}$. □

Proof of Lemma E.7. Fix any agent i P rms. We will argue that the policy $P^{gen;d}_{i\ e}$ t defined within the proof of Lemma E.3 satisfies V_i ¥ m a logpT q{H. Since is an -CCE of G, it follows that

$$¥ V_i V_i \ ¥ m \ a \log T \quad p \quad q{} \quad H V_i \ ;$$

from which the result of Lemma E.7 follows after rearranging terms. To simplify notation, let us write $p_{i;h}$: $q_{j;h}$ p where we recall that each $q_{j;h}$ is determined given the history up to step h, $p_{j;h1}$;s_hq, as defined in (15) and (16). An action profile

drawn from $\rho_{i;h}$ is denoted by $a_i \sim \rho_{i;h}$, with $a_i \in A_i$. We compute

$$
\begin{aligned}
V_i^{\pi_i,\bar{\pi}_{-i}} &= \mathbb{E}_{s_1\sim\rho}\Big[\sum_{h=1}^{H} \mathbb{E}_{a_h\sim\pi_h(\cdot|s_h)}\big[R_{i;h}(s_h,a_h)\big] \,\Big|\, s_1\Big] \\
&\le \mathbb{E}_{s_1\sim\rho}\Big[\sum_{h=1}^{H} \mathbb{E}_{a_h\sim\bar{\pi}_h(\cdot|s_h)}\big[R_{i;h}(s_h,a_h)\big] \,\Big|\, s_1\Big] + \frac{1}{H}\sum_{j\neq i} D_{TV}\big(\rho_{j;h}(s_h),\rho_{j;h+1}(s_h,q_{j;h})\big) \\
&\le \mathbb{E}_{s_1\sim\rho}\Big[\sum_{h=1}^{H} \max_{a_i\in A_i} \mathbb{E}_{a_{-i}\sim\bar{\pi}_{-i;h}}\big[R_{i;h}(s_h,a_i,a_{-i})\big] \,\Big|\, s_1\Big] + \frac{m}{H}\sqrt{\frac{a}{H}\log T} \\
&\le \mathbb{E}_{s_1\sim\rho}\Big[\sum_{h=1}^{H} \mathbb{E}_{a_h\sim\bar{\pi}_h(\cdot|s_h)}\big[R_{i;h}(s_h,a_h)\big] \,\Big|\, s_1\Big] + m\sqrt{\frac{a}{H}\log(T)\log(H)}.
\end{aligned}
$$

where:

- The first inequality follows from the fact that the rewards $R_{i;h}(s,a)$ take values in $[0,1]$ and that the total variation between product distributions is bounded above by the sum of total variation distances between each of the pairs of component distributions.
- The second inequality follows from the definition of $\pi_{i;h}(s_h)$ in terms of $q_{i;h}$ in (17) as well as (20) applied to each $j \neq i$ and each $t \in [T]$.
- The final inequality follows by Lemma E.8 below, applied to agent i and to the distribution $\rho_{i;h}$, which we recall is a product distribution almost surely.

□

Lemma E.8. For any $i \in [m]$, $s \in [S]$, $h \in [H]$, and any product distribution $q \in \prod_{i=1}^{m} A_i$, it holds that

$$\max_{a\in A_i} \mathbb{E}_{a_{-i}\sim q_{-i}}\big[R_{i;h}(s,a,a_{-i})\big] - \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a)\big] \ge 0.$$

Proof. Choose $a_i^\star := \operatorname{argmax}_{a\in A_i} \mathbb{E}_{a_{-i}\sim q_{-i}}[R_{i;h}(s,a,a_{-i})]$. Now we compute

$$
\begin{aligned}
\mathbb{E}_{a\sim q}\big[R_{i;h}(s,a)\big] &\ge \min_{a_{-i}\in A_{-i}} \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a_i^\star,a_{-i})\big] \\
&\ge \min_{j\neq i}\min_{a_j\in A_j} \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a_i^\star,a_j,a_{-ij})\big] \\
&\ge \min_{j\neq i}\min_{a_j\in A_j} \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a_i^\star,a_j,a_{-ij})\big] - \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a_i^\star,a_{-i})\big] + \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a_i^\star,a_{-i})\big] \\
&\ge \mathbb{E}_{a\sim q}\big[R_{i;h}(s,a_i^\star,a_{-i})\big] - 1.
\end{aligned}
$$

where the first inequality follows since q is a product distribution, the second inequality uses that $\mathrm{enc}(p,q)$ is non-negative, and the final inequality follows since by choice of $a_i^\star$ we have $\mathbb{E}_{a\sim q}[R_{i;h}(s,a_i^\star,a_{-i})] \ge \mathbb{E}_{a\sim q}[R_{i;h}(s,a_j,a_{-ij})] - 1$ for all $a_j \in A_j$. □

E.2. Remarks on bit complexity of the rewards

The Markov game $G(G)$ constructed to prove Theorem E.1 uses lower-order bits of the rewards to record the action profile taken each step. These lower order bits may be used by each agent to infer what actions were taken by other agents at the previous step, and we use this idea to construct the best-response policies $\pi_i^\star$ defined in the proof. As a result of this aspect of the construction, the rewards of the game $G(G)$ each take $O(m\log(pnq) + \log(1/\varepsilon))$ bits to specify. As discussed in the proof of Theorem E.1, it is without loss of generality to assume that the payoffs of the given normal-form game G take $O(\log(1/\varepsilon))$ bits each to specify, so when either $m > 1$ or $n > 1/\varepsilon$, the construction of $G(G)$ uses more bits to express its rewards than what is used for the normal-form game G.

It is possible to avoid this phenomenon by instead using the state transitions of the Markov game to encode the action profile taken at each step, as was done in the proof of Theorem 3.2. The idea, which we sketch here, is to replace the game $G(G)$ of Definition E.2 with the following game $G'(G)$:

Definition E.9 (Alternative construction to Definition E.2). Given an m-player, n_0-action normal-form game G, we define the game $G^1(G, H, (A_i)_{i\in[m]}, P, (R_i)_{i\in[m]})$ as follows.

- The horizon of G is $H = n_0$.
- Let $A = n_0$. The action spaces of agents $1, 2, \ldots, m$ are given by $A_1 = \cdots = A_m = [A]$. The action space of agent $m+1$ is A_{m+1}

$$= \{(j, a_j) : j \in [m], a_j \in A_j\},$$

so that $|A_{m+1}| = Am = n$.

We write $\bar{A} = \prod_{j=1}^{m} A_j$ to denote the joint action space of the first m agents, and $\bar{\bar{A}} = \prod_{j=1}^{m+1} A_j$ to denote the joint action space of all agents. Then $|\bar{A}| = A^m = mA^{m-1} = n$.

- The state space S is defined as follows. There are $|\bar{A}|$ states, one for each action tuple $a \in \bar{A}$. For each $a \in \bar{A}$, we denote the corresponding state by s_a.
- For all $h \in [H]$, the reward to agent $j \in [m+1]$ is given action profile $a = (a_1, \ldots, a_{m+1})$ at any state $s \in S$ is as follows: writing $a_{m+1} = (j^1, a^1_{j^1})$,

$$R_{j,h}(s, a) = \begin{cases} 0 & : j \notin \{j^1, m+1\} \\ \frac{1}{H} \cdot M_j(a_1, \ldots, a_m, M_j(a^1_1, \ldots, a^1_m, \ldots, a_m)) & : j = j^1 \\ \frac{1}{H} \cdot M_j(a^1_1, \ldots, a^1_m, \ldots, a_m, M_j(a_1, \ldots, a_m)) & : j = m+1. \end{cases} \tag{23}$$

- At each step $h \in [H]$, if action profile $a \in \bar{A}$ is taken, the game transitions to the state s_a.

Note that the number of states of $G^1(G)$ is equal to $|\bar{A}| \cdot n^{m+1}$, and so $|G^1(G)| = n^{m+1}$. As a result, if we were to use the game $G^1(G)$ in place of $G(G)$ in the proof of Theorem E.1, we would need to define $n_0 := \lceil n^{1/(m+1)} \rceil$ to ensure that $|G^1(G)| = n$, and so the condition $T \le \exp(2^{\lceil n/m \rceil})$ would be replaced by $T \le \exp(2^{\lceil n^{1/(m+1)} \rceil})$. This would only lead to a small quantitative degradement in the statement of Theorem 4.3, with the condition in the statement replaced by $T \le \exp(c \cdot 2^{n^{1/3}})$ for some constant $c > 0$. However, it would render the statement of Theorem 5.2 essentially vacuous. For this reason, we opt to go with the approach of Definition E.2 as opposed to Definition E.9.

We expect that the construction of Definition E.2 can nevertheless still be modified to use $O(\log(1/\varepsilon))$ bits to express each reward in the Markov game G. In particular, one could introduce stochastic transitions to encode in the state of the Markov game a small number of random bits of the full action profile played at each step. We leave such an approach for future work.

F. Equivalence between $\Pi^{\mathrm{gen,rnd}}_j$ and $\Delta(\Pi^{\mathrm{gen,det}}_j)$

In this section we consider an alternate definition of the space $\Pi^{\mathrm{gen,rnd}}_i$ of randomized general policies of player i, and show that it is equivalent to the one we gave in Section 2.

In particular, suppose we were to define a randomized general policy of agent i as a distribution over deterministic general policies of agent i: we write $\Pi^{\mathrm{gen,rnd}}_i := \Delta(\Pi^{\mathrm{gen,det}}_i)$ to denote the space of such distributions. Moreover, write $\Pi^{\mathrm{gen,rnd}} := \Pi^{\mathrm{gen,rnd}}_1 \times \cdots \times \Pi^{\mathrm{gen,rnd}}_m = \Delta(\Pi^{\mathrm{gen,det}}_1) \times \cdots \times \Delta(\Pi^{\mathrm{gen,det}}_m)$ to denote the space of product distributions over agents' deterministic policies. Our goal in this section is to show that policies in $\Pi^{\mathrm{gen,rnd}}$ are equivalent to those in $\Pi^{\mathrm{gen,rnd}}$ in the following sense: there is an embedding map $\mathrm{Emb} : \Pi^{\mathrm{gen,rnd}} \to \Pi^{\mathrm{gen,rnd}}$, not depending on the Markov game, so that the distribution of a trajectory drawn from any $\pi \in \Pi^{\mathrm{gen,rnd}}$, for any Markov game, is the same as the distribution of a trajectory drawn from $\mathrm{Emb}(\pi)$ (Fact F.2). Furthermore, Emb is surjective in the following sense: any policy $\pi \in \Pi^{\mathrm{gen,rnd}}$ produces trajectories that are distributed identically to those of $\mathrm{Emb}(\pi)$ (and thus of π), for some $\pi \in \Pi^{\mathrm{gen,rnd}}$ (Fact F.3). In Definition F.1 below, we define Emb.

Definition F.1. For $j \in [m]$ and $\pi_j \in \Pi^{\mathrm{gen,rnd}}_j$, define $\mathrm{Emb}_j(\pi_j) \in \Pi^{\mathrm{gen,rnd}}_j = \Delta(\Pi^{\mathrm{gen,det}}_j)$ to put the following amount of mass on each $\pi_j \in \Pi^{\mathrm{gen,det}}_j$:

$$\mathrm{Emb}_j(\pi_j)(\pi_j) := \prod_{h=1}^{H} \prod_{(s_{j,h+1}, s_h) \in \mathrm{Pre}_{j,h+1}(S)} \pi_{j,h}(\pi_{j,h+1}, s_h) \mid_{j,h+1}(s_{j,h+1}, s_h) \tag{24}$$

Furthermore, for $p_1,\ldots,p_m \in \Delta^{\mathrm{gen;rnd}}$, define $\mathrm{Emb}(p_1,\ldots,p_m) = (\mathrm{Emb}(p_1),\ldots,\mathrm{Emb}(p_m))$.

Note that, in the special case that $p_j \in \Delta^{\mathrm{gen;det}}_j$, $\mathrm{Emb}_j(p_j)$ is the point mass on p_j.

Fact F.2 (Embedding equivalence). Fix a m-player Markov game G and, arbitrary policies $p_j \in \Delta^{\mathrm{gen;rnd}}_j$. Then a trajectory drawn from the product policy $(p_1,\ldots,p_m) \in \Delta^{\mathrm{gen;rnd}}_1 \times \cdots \times \Delta^{\mathrm{gen;rnd}}_m$ is distributed identically to a trajectory drawn from $\mathrm{Emb}(p) \in \Delta^{\mathrm{gen;rnd}}$.

The proof of Fact F.2 is provided in Section F.1. Next, we show that the mapping Emb is surjective in the following sense:

Fact F.3 (Right inverse of Emb_j). There is a mapping $\mathrm{Fac} : \Delta^{\mathrm{gen;rnd}} \to \Delta^{\mathrm{gen;rnd}}$ so that for any Markov game G and any $r \in \Delta^{\mathrm{gen;rnd}}$, the distribution of a trajectory drawn from r is identical to the distribution of a trajectory drawn from $\mathrm{Emb}(\mathrm{Fac}(r))$.

We will write $\mathrm{Fac}(r_1,\ldots,r_m) : (\mathrm{Fac}_1(r_1),\ldots,\mathrm{Fac}_m(r_m))$. Fact F.3 states that the policy $\mathrm{Fac}(r)$ maps, under Emb, to a policy in $\Delta^{\mathrm{gen;rnd}}$ which is equivalent to r (in the sense that their trajectories are identically distributed for any Markov game).

An important consequence of Fact F.2 is that the expected reward (i.e., value) under any $p \in \Delta^{\mathrm{gen;rnd}}$ is the same as that of $\mathrm{Emb}(p)$. Thus given a Markov game, the induced normal-form game in which the players' pure action sets are $\Delta^{\mathrm{gen;rnd}}_1,\ldots,\Delta^{\mathrm{gen;rnd}}_m$ is equivalent to the normal-form game in which the players' pure action sets are $\Delta^{\mathrm{gen;det}}_1,\ldots,\Delta^{\mathrm{gen;det}}_m$, in the following sense: for any mixed strategy in the former, namely a product distributional policy $P = P(p^{\mathrm{gen;rnd}}) \in \Delta(\Delta^{\mathrm{gen;rnd}})$, the policy $E_P[\mathrm{Emb}(p)] \in \Delta(\Delta^{\mathrm{gen;det}}_1) \times \cdots \times \Delta(\Delta^{\mathrm{gen;det}}_m)$ is a mixed strategy in the latter which gives each player the same value as under P. (Note that $E_P[\mathrm{Emb}(p)]$ is indeed a product distribution since P is a product distribution and Emb factors into individual coordiantes.) Furthermore, by Fact F.3, any distributional policy in $\Delta^{\mathrm{gen;rnd}}$ arises in this manner, for some $P = P(p^{\mathrm{gen;rnd}}_1) \times \cdots \times \Delta(p^{\mathrm{gen;rnd}}_m)$; in fact, P may be chosen to place all its mass on a single $p \in \Delta^{\mathrm{gen;rnd}}_1 \times \cdots \times \Delta^{\mathrm{gen;rnd}}_m$. Since Emb factors into individual coordinates, it follows that Emb yields a one-to-one mapping between the coarse correlated equilibria (or any other notion of equilibria, e.g., Nash equilibria or correlated equilibria) of these two normal-form games.

F.1. Proofs of the equivalence

Proof of Fact F.2. Consider any trajectory $(s_1,a_1,r_1,\ldots,s_H,a_H,r_H)$ consisting of a sequence of H states and actions and rewards for each of the m agents. Assume that $r_{i,h} \in R_{i,h}(s,a_h)$ for all i,h (as otherwise has probability 0 under any policy). Write:

$$p : \prod_{h=1}^{H-1} P_h(s_{h+1} \mid s_h, a_h)(r_h)$$

Then the probability of observing under is

$$p \prod_{h=1}^{H} \prod_{j=1}^{m} \pi_{j,h}(a_{j,h} \mid \tau_{j,h-1}, s_h) \tag{25}$$

where, per usual, $\tau_{j,h-1} = (s_1,a_{j,1},r_{j,1},\ldots,s_{h-1},a_{j,h-1},r_{j,h-1})$. Write $(p_1,\ldots,p_m) = \mathrm{Emb}(p)$. The probability of observing under is

$$p \prod_{j=1}^{m} \Pr_{\sigma_j \sim p_j}[\sigma_j : \forall h, \sigma_j(\tau_{j,h-1}, s_h) = a_{j,h}] \tag{26}$$

It is now straightforward to see from the definition of $\pi_j(p_j)$ in (24) that the quantities in (25) and (26) are equal. □

Proof of Fact F.3. Fix a policy $r_j \in \Delta(\Delta^{\mathrm{gen;rnd}}_j) = \Delta(\Delta^{\mathrm{gen;det}}_j)$. We define $\mathrm{Fac}_j(r_j)$ to be the policy $p_j \in \Delta^{\mathrm{gen;rnd}}_j$, which is defined as follows: for $\tau_{j,h-1} = (s_{j,1},a_{j,1},r_{j,1},\ldots,s_{j,h-1},a_{j,h-1},r_{j,h-1}) \in H_{j,h-1}$, $s_h \in S$, we have, for $a_{j,h} \in A_j$,

$$p_j(a_{j,h} \mid \tau_{j,h-1}, s_h) = \frac{\sum_{p_j \in \Delta^{\mathrm{gen;det}}_j} r_j(p_j)\, p_j(a_{j,g} \mid \tau_{j,g-1}, s_g)}{\sum_{p_j \in \Delta^{\mathrm{gen;det}}_j} r_j(p_j)\, p_j(a_{j,g} \mid \tau_{j,g-1}, s_g)}.$$

If the denominator of the above expression is 0, then $\pi_j(\cdot \mid j;h-1;s_h)$ is defined to be an arbitrary distribution on $\Delta(A_j)$. (For concreteness, let us say that it puts all its mass on a fixed action in A_j.) Furthermore, for $\vec{r} \in \mathcal{R}^{gen;rnd}$, define $Fac_{\vec{r}}(\vec{\pi}) : (Fac_1(\pi_1), \ldots, Fac_m(\pi_m)) \in \Pi^{gen;rnd}$.

Next, fix any $\vec{r} = (r_1, \ldots, r_m) \in \mathcal{R}_1^{gen;rnd} \times \cdots \times \mathcal{R}_m^{gen;rnd}$. Let $\vec{\pi} = Fac_{\vec{r}}(\vec{\rho})$. By Fact F.2, it suffices to show that the distribution of trajectories under $\vec{\rho}$ is the same as the distribution of trajectories drawn from $\vec{\pi}$.

So consider any trajectory $(s_1, a_1, r_1, \ldots, s_H, a_H, r_H)$ consisting of a sequence of H states and actions and rewards for each of the m agents. Assume that $r_{i,h} = R_{i,h}(s, a_h)$ for all i, h (as otherwise has probability 0 under any policy). Write:

$$p := \prod_{h=1}^{H} P_h(s_{h+1} \mid s_h, a_h)$$

Then the probability of observing under $\vec{\rho}$ is

$$p \prod_{h=1}^{H} \prod_{j=1}^{m} \rho_{j,h}(a_{j,h} \mid o_{j,h-1}, s_h)$$

$$= p \prod_{h=1}^{H} \prod_{j=1}^{m} \frac{r_j(s_h, a_h) \cdot \left(\rho^{gen,det}_j(o, s, a)\right)}{\sum_{g=1}^{m} r_g(s_h, a_h) \cdot \left(\rho^{gen,det}_g(o, s, a)\right) + \sum_{j=1}^{m} r_j(s_h, a_h) \cdot \left(\rho^{gen,det}_j(o, s, a)\right)}$$

$$= p \prod_{h=1}^{H} \frac{\sum_{j=1}^{m} r_j(s_h, a_h) \cdot \left(\rho^{gen,det}_j(o, s, a)\right)}{\sum_{j=1}^{m} r_j(s_h, a_h) \cdot \left(\rho^{gen,det}_j(o, s, a)\right)};$$

which is equal to the probability of observing under $\vec{r}$. □