Solving Discounted Stochastic Two-Player Games with Near-Optimal Time and Sample Complexity

Aaron Sidford
Stanford University
sidford@stanford.edu

Mengdi Wang
Princeton University
mengdiw@princeton.edu

Lin F. Yang
UCLA
linyang@ee.ucla.edu

Yinyu Ye
Stanford University
yyye@stanford.edu

Abstract

In this paper we settle the sampling complexity of solving discounted two-player turn-based zero-sum stochastic games up to polylogarithmic factors. Given a stochastic game with discount factor $\gamma \in (0, 1)$ we provide an algorithm that computes an ϵ-optimal strategy with high-probability given $\widetilde{O}((1-\gamma)^{-3}\epsilon^{-2})$ samples from the transition function for each state-action-pair. Our algorithm runs in time nearly linear in the number of samples and uses space nearly linear in the number of state-action pairs. As stochastic games generalize Markov decision processes (MDPs) our runtime and sample complexities are optimal due to Azar et al. (2013). We achieve our results by showing how to generalize a near-optimal Q-learning based algorithms for MDP, in particular Sidford et al. (2018a), to two-player strategy computation algorithms. This overcomes limitations of standard Q-learning and strategy iteration or alternating minimization based approaches and we hope will pave the way for future reinforcement learning results by facilitating the extension of MDP results to multi-agent settings with little loss.

1 Introduction

In this paper we study the sample complexity of learning a near-optimal strategy in discounted two-player turn-based zero-sum stochastic games Shapley (1953); Hansen et al. (2013), which we refer to more concisely as *stochastic games*. Stochastic games model dynamic strategic settings in which two players take turns and the state of game evolves stochastically according to some transition law. This model encapsulates a major challenge in multi-agent learning: other agents may be learning and adapting as well. Further, stochastic games are a generalization of the Markov decision process (MDP), a fundamental model for reinforcement learning, to the two-player setting Littman (1994). MDPs can be viewed as degenerate stochastic games in which one of the players has no influence. Consequently, understanding stochastic games is a natural step towards resolving challenges in reinforcement learning of extending single-agent learning to multi-agent settings.

There is a long line of research in both MDPs and stochastic games (for a more thorough introduction, see Filar and Vrieze (2012); Hansen et al. (2013) and references therein). Strikingly, Hansen et al. (2013) showed that there exists a pure-strategy Nash equilibrium which can be computed in strongly polynomial time for stochastic games, if the game matrix is fully accessible and the discount factor is fixed. In reinforcement learning settings, however, the transition function of the game is unknown and a common goal is to obtain an approximately optimal strategy (a function that maps states to actions) that is able to obtain an expected cumulative reward of at least (or at most) the Nash equilibrium value no matter what the other player does. Unfortunately, despite interest in generalizing MDP results to stochastic games, currently the best known running times/sample complexity for solving stochastic games in a variety of settings are worse than for solving MDPs. This may not be surprising since in general stochastic games are harder to solve than MDPs, e.g., whereas MDPS can be solved in (weakly) polynomial time it remains open whether or not the same can be done for stochastic games.

There are two natural approaches towards achieving sample complexity bounds for solving stochastic games. The first is to note that the popular stochastic value iteration, dynamic programming, and Q-learning methods all apply to stochastic games Littman (1994); Hu and Wellman (2003); Littman (2001a); Perolat et al.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

(2015). Consequently, recent advances in these methods Kearns and Singh (1999); Sidford et al. (2018b) developed for MDPs can be directly generalized to solving stochastic games (though the sample complexity of these generalized methods has not been analyzed previously). It is tempting to generalize the analysis of sample optimal methods for estimating values Azar et al. (2013) and estimating policies Sidford et al. (2018a) of MDPs to stochastic games. However, this is challenging as these methods rely on monotonicities in MDPs induced by the linear program nature of the problem Azar et al. (2013); Sidford et al. (2018a).

The second approach would be to apply strategy iteration or alternating minimization / maximization to reduce solving stochastic games to approximately solving a sequence of MDPs. Unfortunately, the best analysis of such a method Hansen et al. (2013) requires solving $\Omega(1/(1-\gamma))$ MDPs. Consequently, even if this approach could be carried out with approximate MDP solvers, the resulting sample complexity for solving stochastic games would be larger than that needed for solving MDPs. More discussion of related literatures is given in Section 1.4.

Given the importance of solving stochastic games in reinforcement learning (e.g. Hu et al. (1998); Bowling and Veloso (2000, 2001); Hu and Wellman (2003); Arslan and Yüksel (2017)), this suggests the following fundamental open problem:

Can we design stochastic game learning algorithms that provably match the performance of MDP algorithms and achieve near-optimal sample complexities?

In this paper, we answer this question in the affirmative in the particular case of solving discounted stochastic games with a generative model, i.e. an oracle for sampling from the transition function for state-action pairs. We provide an algorithm with the same near-optimal sample complexity that is known for solving discounted MDPs. Further, we achieve this result by showing how to transform particular MDP algorithms to solving stochastic games that satisfy particular two-sided monotonicity constraints. Therefore, while there is a major gap between MDPs and stochastic games in terms of computation time for obtaining the exact solutions, this gap disappears when considering the sampling complexity between the two. We hope this work opens the door to more generally extend results for MDP to stochastic games and thereby enable the application of the rich research on reinforcement learning to a broader multi-player settings with little overhead.

1.1 The Model

Formally, throughout this paper, we consider *discounted turn-based two-player zero-sum stochastic games* described as the tuple $\mathcal{G} = (\mathcal{S}_{\min}, \mathcal{S}_{\max}, \mathcal{A}, \boldsymbol{P}, \boldsymbol{r}, \gamma)$. In these games there are two players, a *min* or *minimization* player which seeks to minimize the cumulative reward in the game and a *max* or *maximization* player which seeks to maximize the cumulative reward. Here, $\mathcal{S}_{\min}$ and $\mathcal{S}_{\max}$ are disjoint finite sets of *states* controlled by the min-player and the max-player respectively and their union $\mathcal{S} := \mathcal{S}_{\min} \cup \mathcal{S}_{\max}$ is the set of all possible *states of the game*. Further, $\mathcal{A}$ is a finite set of *actions* available at each state, $\boldsymbol{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$ is a *transition probability function*, $\boldsymbol{r} : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ is the payoff or *reward function* and $\gamma \in (0,1)$ is a discount factor.[1]

Stochastic games $\mathcal{G} = (\mathcal{S}_{\min}, \mathcal{S}_{\max}, \mathcal{A}, \boldsymbol{P}, \boldsymbol{r}, \gamma)$ are played dynamically in a sequence of turns, $\{t\}_{t=0}^{\infty}$, starting from some initial state $s^0 \in \mathcal{S}$ at turn $t = 0$. In each turn $t \geq 0$, the game is in one of the states $s^t \in \mathcal{S}$ and the player who controls the state s^t chooses or *plays* an action a^t from the action space $\mathcal{A}$. This action yields reward $r^t := r(s^t, a^t)$ for the turn and causes the next state s^{t+1} to be chosen at random from $\mathcal{S}$ where the transition probability $\Pr[s^{t+1} = s' | s_1, ..., s_t, a_1, ..., a_t] = \boldsymbol{P}(s' \mid s^t, a^t)$. The goal of the min-player (resp. max-player) is to choose actions to minimize (resp. maximize) the expected infinite-horizon discounted-reward or *value* of the game $\sum_{t=0}^{\infty} \gamma^t r^t$.

In this paper we focus on the case where the players play pure (deterministic) stationary strategies (policies), i.e. strategies which depend only on the current state. That is we wish to compute a *min-player strategy* or *policy* $\pi_{\min} : \mathcal{S}_{\min} \to \mathcal{A}$ which defines the action the min player chooses at a state in $\mathcal{S}_{\min}$ and *max-player strategy* $\pi_{\max} : \mathcal{S}_{\max} \to \mathcal{A}$ which defines the action the max player chooses at a state in $\mathcal{S}_{\max}$. We call a pair of min-player and max-player strategies $\sigma = (\pi_{\min}, \pi_{\max})$ simply a *strategy*. Further, we let $\sigma(s) := \pi_{\min}(s)$ for $s \in \mathcal{S}_{\min}$ and $\sigma(s) := \pi_{\max}(s)$ for $s \in \mathcal{S}_{\max}$ and define the *value function* or *expected discounted cumulative reward* by $\boldsymbol{v}^{\sigma}$ where

$$\boldsymbol{v}^{\sigma}(s) = \boldsymbol{v}[\sigma](s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}(s^t, \sigma(s^t)) \;\middle|\; s^0 = s\right] \quad \text{for all } s \in \mathcal{S}$$

and the expectation is over the random sequence of states, $s^0, s^1, s^2, \ldots$ generated according to $\boldsymbol{P}$ under the strategy σ, i.e. $\Pr[s^{t+1} = s' \mid s^t, s^{t-1}, ..., s^0] = \boldsymbol{P}(s' \mid s^t, \sigma(s^t))$ for all $t > 0$.

[1] Standard reductions allow this result to be applied for rewards of a broader range Sidford et al. (2018a). Further, while we assume there are the same number of actions per-state, our results easily extend to the case where this is non-uniform; in this case our dependencies on $|\mathcal{S}||\mathcal{A}|$ can be replaced with the number of state-action pairs.

Our goal in solving a game is to compute an approximate *Nash equilibrium* restricted to stationary strategies Nash (1951); Maskin and Tirole (2001). We call a strategy $\sigma = (\pi_{\min}, \pi_{\max})$ an *equilibrium strategy* or *optimal* if

$$\max_{\pi'_{\max}:\mathcal{S}_{\max}\to\mathcal{A}} \boldsymbol{v}^{(\pi_{\min},\pi'_{\max})} \le \boldsymbol{v}^{\sigma} \le \min_{\pi'_{\min}:\mathcal{S}_{\min}\to\mathcal{A}} \boldsymbol{v}^{(\pi'_{\min},\pi_{\max})}.$$

and we call it ϵ-optimal if these same inequalities hold up to an additive ϵ entrywise. It is worth noting that the best response strategy to a stationary policy is also stationary Fudenberg and Tirole (1991) and there always exists a pure stationary strategy attaining the Nash equilibrium Shapley (1953). Consequently, it is sufficient to focus on deterministic strategies.

Throughout this paper we focus on solving stochastic games in the learning setting where the game is not fully specified. We assume that a *generative model* is available which given any state-action pair, i.e. $s \in \mathcal{S}$ and $a \in \mathcal{A}$, can sample a random s' independently at random from the transition probability function, i.e. $\Pr[s' = t] = \boldsymbol{P}(t \mid s, a)$. Accessibility to a generative model is a standard and natural assumption (Kakade (2003); Azar et al. (2013); Sidford et al. (2018a); Agarwal et al. (2019)) and corresponds to PAC learning. The special case of solving a MDP given a generative model has been studied extensively (Kakade (2003); Azar et al. (2013); Sidford et al. (2018b,a); Agarwal et al. (2019)) and is a natural proving ground towards designing theoretically motivated reinforcement learning algorithms.

1.2 Our Results

In this paper we provide an algorithm that computes an ϵ-optimal strategy using a sample size that matches the best known sample complexity for solving discounted MDPs. Further, our algorithm runs in time proportional to the number of samples and space proportional to $|\mathcal{S}||\mathcal{A}|$. Interestingly, we achieve this result by showing how to run two-player variant of Q-learning such that the value-strategy sequences induced enjoy certain monotonicity properties. Essentially, we show that provided a value improving algorithm is sufficiently stable, then it can be extended to the two-player setting with limited loss. This allows us to leverage recent advances in solving single player games to solve stochastic games with limited overhead. Our main result is given below.

Theorem 1.1 (Main Theorem)**.** *There is an algorithm which given a stochastic game, $\mathcal{G} = (\mathcal{S}_{\min}, \mathcal{S}_{\max}, \boldsymbol{P}, \boldsymbol{r}, \gamma)$ with a generative model, outputs, with probability at least $1-\delta$, an ϵ-optimal strategy σ by querying $Z = \widetilde{O}(|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-3}\epsilon^{-2})$ samples, where $\epsilon \in (0,1)$ and $\widetilde{O}(\cdot)$ hides polylogarithmic factors. The algorithm runs in time $O(Z)$ and uses space $O(|\mathcal{S}||\mathcal{A}|)$.*

Our sample and time complexities are optimal due to a known lower bound in the single player case by Azar et al. (2013). It was shown in Azar et al. (2013) that solving any one-player MDP to ϵ-optimality with high probability needs at least $\Omega(|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-3}\epsilon^{-2})$ samples. Our sample complexity upper bound generalizes the recent sharp sample complexity results for solving the discounted MDP Sidford et al. (2018a); Agarwal et al. (2019), and tightly matches the information-theoretic sample complexity up to polylogarithmic factors. This result provides the first and near-optimal sample complexity for solving the two-person stochastic game.

1.3 Notations and Preliminaries

Notation: We use $\mathbf{1}$ to denote the all-ones vector whose dimension is adapted to the context. We use the operators $|\cdot|, (\cdot)^2, \sqrt{\cdot}, \le, \ge$ as entrywise operators on vectors. We identify the transition probability function $\boldsymbol{P}$ as a matrix in $\mathbb{R}^{(\mathcal{S}\times\mathcal{A})\times\mathcal{S}}$ and each row $\boldsymbol{P}(\cdot \mid s, a) \in \mathbb{R}^{\mathcal{S}}$ as a vector. We denote $\boldsymbol{v}$ as a vector in $\mathbb{R}^{\mathcal{S}}$ and $\boldsymbol{Q}$ as a vector in $\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$. Therefore $\boldsymbol{P}\boldsymbol{v}$ is a vector in $\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$. We use σ to denote strategy pairs and π for the min-player or max-player strategy. For any strategy σ, we define $\boldsymbol{Q}_\sigma \in \mathbb{R}^{\mathcal{S}}$ as $\boldsymbol{Q}_\sigma(s) := \boldsymbol{Q}(s, \sigma(s))$ for $\forall s \in \mathcal{S}$. We denote $\boldsymbol{P}^\sigma$ as a linear operator defined as

$$\forall s \in \mathcal{S}: \quad [\boldsymbol{P}^\sigma \boldsymbol{v}](s) = \boldsymbol{P}(\cdot \mid s, \sigma(s))^\top \boldsymbol{v},$$
$$\forall s, a \in \mathcal{S}\times\mathcal{A}: [\boldsymbol{P}^\sigma \boldsymbol{Q}](s,a) = \boldsymbol{P}(\cdot \mid s, a)^\top \boldsymbol{Q}_\sigma.$$

Min-value and max-value: For a min-player strategy $\pi_{\min}$, we define its *value* as

$$\boldsymbol{v}^{\pi_{\min}} := \max_{\pi_{\max}\,:\,\mathcal{S}_{\max}\to\mathcal{A}} \boldsymbol{v}^{(\pi_{\min},\pi_{\max})}, \tag{1}$$

We let $\sigma_{\max}(\pi_{\min})$ denote a maximizing argument of the above and call it an *optimal counter strategy* of $\pi_{\min}$. Thus a value of a min-player strategy gives his expected reward in the *worst case*. We say a min-player strategy $\pi_{\min}$ is ϵ-*optimal* if

$$\boldsymbol{v}^{\pi_{\min}} \le \min_{\pi'_{\min}\,:\,\mathcal{S}_{\min}\to\mathcal{A}} \boldsymbol{v}^{\pi'_{\min}} + \epsilon\cdot\mathbf{1}, \quad \text{entrywisely.}$$

The value and ϵ-optimality for the max player is defined similarly. We denote by σ^* the optimal strategy and by $\boldsymbol{v}^*$ the value function of the optimal strategy.

Q-function: For a strategy σ, we denote its Q-*function* (or *action value*) as $\boldsymbol{Q}^\sigma \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ by $\boldsymbol{Q}^\sigma := \boldsymbol{r} + \gamma\boldsymbol{P}\boldsymbol{v}^\sigma$. For a vector $\boldsymbol{v} \in \mathbb{R}^{\mathcal{S}}$ we denote $\boldsymbol{Q}(\boldsymbol{v}) := \boldsymbol{r} + \gamma\boldsymbol{P}\boldsymbol{v}$. Given a $\boldsymbol{Q} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, we denote the greedy value of $\boldsymbol{Q}$ as

$$V[\boldsymbol{Q}](s) := \min_{a\in\mathcal{A}} \boldsymbol{Q}(s,a) \quad \text{if} \quad s \in \mathcal{S}_{\min}$$
$$\text{and} \quad V[\boldsymbol{Q}](s) := \max_{a\in\mathcal{A}} \boldsymbol{Q}(s,a) \quad \text{if} \quad s \in \mathcal{S}_{\max}.$$

Bellman Operator: We denote the Bellman operator, $\mathcal{T}$, as follows: $\mathcal{T}[\boldsymbol{v}] \in \mathbb{R}^{\mathcal{S}}$, and

$$\mathcal{T}[\boldsymbol{v}](s) := V[\boldsymbol{r} + \gamma \boldsymbol{P}\boldsymbol{v}].$$

We also denote the greedy strategy, $\sigma(\boldsymbol{v})$ or $\sigma(\boldsymbol{Q})$, as the maximization/minimization argument of the $\mathcal{T}$ operator. Moreover, for a given strategy σ, we denote $\mathcal{T}_\sigma[\boldsymbol{v}] = \boldsymbol{Q}(\boldsymbol{v})_\sigma$. For a given min-player strategy $\pi_{\min}$, we define the *half* Bellman operator $\mathcal{H}_{\pi_{\min}}$

$$\mathcal{H}_{\pi_{\min}}[\boldsymbol{v}](s) = \boldsymbol{r}(s, \pi_{\min}(s)) + \gamma \boldsymbol{P}(\cdot \mid s, \pi_{\min}(s))^\top \boldsymbol{v} \quad \text{if} \quad s \in \mathcal{S}_{\min};$$
$$\mathcal{H}_{\pi_{\min}}[\boldsymbol{v}](s) \quad \text{if} \quad s \in \mathcal{S}_{\max}.$$

We define $\mathcal{H}_{\pi_{\max}}$ similarly. Note that $\boldsymbol{v}^*$ is the unique fixed point of the Bellman operator, i.e., $\mathcal{T}[\boldsymbol{v}^*] = \boldsymbol{v}^*$ (known as the Bellman equation Bellman (1957)). Similarly, $\boldsymbol{v}^{\pi_{\min}}$ (resp. $\boldsymbol{v}^{\pi_{\max}}$) is the unique fixed point for $\mathcal{H}_{\pi_{\min}}$ (resp. $\mathcal{H}_{\pi_{\max}}$). The (half) Bellman-operators satisfy the following properties (see. e.g. Hansen et al. (2013); Puterman (2014))

1. *contraction*: $\|\mathcal{T}[\boldsymbol{v}_1] - \mathcal{T}[\boldsymbol{v}_2]\|_\infty \leq \gamma \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|_\infty$;
2. *monotonicity*: $\boldsymbol{v}_1 \leq \boldsymbol{v}_2 \Rightarrow \mathcal{T}[\boldsymbol{v}_1] \leq \mathcal{T}[\boldsymbol{v}_2]$.

High Probability: we say an algorithm has a property "with high probability" if for any δ by increasing the time and sample complexity by $O(\log(1/\delta))$ it has the property with probability $1 - \delta$.

1.4 Previous Work

Here we provide a more detailed survey of previous works related to stochastic games and MDPs. Two-person stochastic games generalize MDPs Shapley (1953). When one of the players has only one action to choose from, the problem reduces to a MDP. A related game is the stochastic game where both players choose their respective actions simultaneously at each state and the process transitions to the next state under the control of both players Shapley (1953). The turn-based stochastic game can be reduced to the game with simultaneous moves Pérolat et al. (2015).

Computing an optimal strategy for a two-player turn-based zero-sum stochastic game is known to be in NP $\cap$ co-NP Condon (1992). Later Hansen et al. (2013) showed that the strategy iteration, a generalization of Howard's policy iteration algorithm Howard (1960), solves the discounted problem in strongly polynomial time when the discount factor is fixed. Their work uses ideas from Ye (2011) which proved that the policy iteration algorithm solves the discounted MDP (DMDP) in strongly polynomial time when the discount factor is fixed. In general (e.g., if the discount factor is part of the input size), it is open if stochastic games can be solved in polynomial time Littman (1996). This is in contrast to MDPs which can be solved in (weakly) polynomial time as they are a special case of linear programming.

The algorithms and complexity theory for solving two-player stochastic games is closely related to that of solving MDPs. Their is vast literature on solving MDPs which dates back to Bellman who developed value iteration in 1957 Bellman (1957). The policy iteration was introduced shortly after by Howard Howard (1960), and its complexity has been extensive studied in Mansour and Singh (1999); Ye (2011); Scherrer (2013). Then d'Epenoux (1963) and De Ghellinck (1960) discovered that MDPs are special cases of a linear program, which leads to the insight that the simplex method, when applied to solving DMDPs, is a simple policy iteration method. Ye Ye (2011) showed that policy iteration (which is a variant of the general simplex method for linear programming) and the simplex method are strongly polynomial for DMDP and terminate in $O(|\mathcal{S}|^2|\mathcal{A}|(1-\gamma)^{-1}\log(|\mathcal{S}|(1-\gamma)^{-1}))$ iterations. Hansen et al. (2013) and Scherrer (2013) improved the iteration bound to $O(|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-1}\log(|\mathcal{S}|(1-\gamma)^{-1}))$ for Howard's policy iteration method. The best known convergence result for policy and strategy iteration are given by Ye (2005) and Hansen et al. (2013). The best known iteration complexities for both problems are of the order $(1-\gamma)^{-1}$, which becomes unbounded as $\gamma \to 1$. It is worth mentioning that Ye (2005) designed a combinatorial interior-point algorithm (CIPA) that solves the DMDP in strongly polynomial time.

Sample-based algorithms for learning value and policy functions for MDP have been studied in Kearns and Singh (1999); Kakade (2003); Singh and Yee (1994); Azar et al. (2011b, 2013); Sidford et al. (2018b,a); Agarwal et al. (2019) and many others. Among these papers, Azar et al. (2013) obtains the first tight sample bound for finding an ϵ-optimal value function and for finding ϵ-optimal policies in a restricted ϵ regime and Sidford et al. (2018a) obtains the first tight sample bound for finding an ϵ-optimal *policy* for any ϵ. Both sample complexities are of the form $\widetilde{O}[|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-3}]$. Lower bounds have been shown in Azar et al. (2011a); Even-Dar et al. (2006) and Azar et al. (2013). Azar et al. (2013) give the first tight lower bound $\Omega[|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-3}]$. For undiscounted average-reward MDP, a primal-dual based method was proposed in Wang (2017) which achieves sample complexity $\widetilde{O}(|\mathcal{S}||\mathcal{A}|t_{\text{mix}}^2 c_{\max}^2/c_{\min}^2)$, where t_{mix} is the worst-case mixing time and $c_{\max}/c_{\min}$ is the ergodicity ratio. Sampling-based method for two-player stochastic game has been considered in Wei et al. (2017) in an online learning setting. However, their algorithm leads to a sub-optimal sample-complexity when generalized to the generative model setting.

As for general stochastic games, the minimax Q-learning algorithm and the friend-and-foe Q-learning algorithm were introduced in Littman (1994) and Littman (2001a), respectively. The Nash Q-learning algorithm was proposed for zero-sum games in Hu and Wellman (2003) and for general-sum games in Littman (2001b); Hu and Wellman (1999).

2 Technique Overview

Since stochastic games are a generalization of MDPs, many techniques for solving MDPs can be immediately generalized to stochastic games. However, as we have discussed, some of the techniques used to achieve optimal sample complexities for solving MDPs in a generative model do not have a clear generalization to stochastic games. Nevertheless, we show how to design an algorithm that carefully extends particular Q-learning based methods, i.e. methods that always maintain an estimator for the optimal value function (or $\boldsymbol{Q}^*$), to achieve our goals.

Q-Learning: To motivate our approach we first briefly review previous Q-learning based methods and the core technique that achieves near-optimal sample complexity. To motivate Q-learning, we first recall the value iteration algorithm solving an MDP. Given a full model for the MDP value iteration updates the iterates as follows

$$\boldsymbol{v}^{(i)} \leftarrow \mathcal{T}[\boldsymbol{v}^{(i-1)}] := V[\boldsymbol{Q}(\boldsymbol{v}^{(i-1)})]$$

where $\boldsymbol{v}^{(0)}$ can be an arbitrary vector. Since the Bellman operator is contractive and $\boldsymbol{v}^*$ is a fix point of $\mathcal{T}$, this method gives an ϵ-optimal value in $O[(1-\gamma)^{-1}\log(\epsilon^{-1})]$ iterations. In the learning setting, $\mathcal{T}$ cannot be exactly computed. The Q-learning approach estimates $\mathcal{T}$ by its approximate version, i.e., to compute $\boldsymbol{P}(\cdot \mid s,a)^\top \boldsymbol{v}^{(i-1)}$, we obtain samples from $\boldsymbol{P}(\cdot \mid s,a)$, and then compute the empirical average. Then we compute the approximate Q-value at the i-th iteration as

$$\boldsymbol{Q}^{(i)} = \widehat{\boldsymbol{Q}}[\boldsymbol{v}^{(i-1)}] := \boldsymbol{r} + \widehat{\boldsymbol{P}}\boldsymbol{v}^{(i-1)}$$
$$\text{and} \quad \widehat{\mathcal{T}}(\boldsymbol{v}^{(i-1)}) := V[\widehat{\boldsymbol{Q}}(\boldsymbol{v}^{(i-1)})],$$

where

$$\widehat{\boldsymbol{P}}(\cdot \mid s,a)^\top \boldsymbol{v} = \frac{1}{m} \sum_{s_i \sim P(\cdot|s,a),\ i\in[m]} \boldsymbol{v}(s_i)$$

for some $m > 0$. Then the estimation error per step is defined as

$$\boldsymbol{\epsilon}^{(i)} = \boldsymbol{Q}[\boldsymbol{v}^{(i-1)}] - \widehat{\boldsymbol{Q}}[\boldsymbol{v}^{(i-1)}].$$

Since the exact value iteration takes at least $\Omega[(1-\gamma)^{-1}]$ iterations to converge, the Q-learning (or approximated value iteration) takes at least $\Omega[(1-\gamma)^{-1}]$ iterations. The total number of samples used over all the iterations is the sample complexity of the algorithm.

Variance Control and Monotonicity Techniques: To obtain the optimal sample complexity for one-player MDP, one approach is to carefully bound each entry of $\boldsymbol{\epsilon}^{(i)}$. By Bernstein inequality (Azar et al. (2013); Sidford et al. (2018a); Agarwal et al. (2019)), we have, with high probability,

$$|\boldsymbol{\epsilon}^{(i)}| \lesssim \sqrt{\mathrm{var}(\boldsymbol{v}^{(i-1)})/m} \leq \sqrt{\mathrm{var}(\boldsymbol{v}^*)/m} + \text{ lower-order terms.}$$

where $\mathrm{var}(\boldsymbol{v}) = \boldsymbol{P}\boldsymbol{v}^2 - (\boldsymbol{P}\boldsymbol{v})^2$ is the *variance-of-value* vector and "$\lesssim$" means "approximately less than." Let $\pi^{(i)}$ be a policy maintained in the i-th iteration (e.g. the greedy policy of the current Q-value). Due to the estimation error $\boldsymbol{\epsilon}^{(i)}$, the per step error bound reads,

$$\boldsymbol{Q}^* - \boldsymbol{Q}^{(i)} \lesssim \gamma \boldsymbol{P}^{\pi^*}\boldsymbol{Q}^* - \gamma \boldsymbol{P}^{\pi^{(i-1)}}\boldsymbol{Q}^{(i-1)} + \boldsymbol{\epsilon}^{(i)}.$$

To derive the overall error accumulation, Sidford et al. (2018a) use the crucial *monotonicity* property, i.e., since $\pi^{(i-1)}(s) = \arg\max_a \boldsymbol{Q}^{(i-1)}(s,a)$, we have

$$\boldsymbol{Q}^{(i-1)}(s,\pi^*(s)) \leq \boldsymbol{Q}^{(i)}(s,\pi^{(i-1)}(s)). \tag{2}$$

We thus have

$$\boldsymbol{Q}^* - \boldsymbol{Q}^{(i)} \lesssim \gamma \boldsymbol{P}^{\pi^*}\boldsymbol{Q}^* - \gamma \boldsymbol{P}^{\pi^*}\boldsymbol{Q}^{(i-1)} + \boldsymbol{\epsilon}^{(i)}.$$

By induction, we have

$$\boldsymbol{Q}^* - \boldsymbol{Q}^{(i)} \leq (I - \gamma \boldsymbol{P}^{\pi^*})^{-1}\sqrt{\mathrm{var}(\boldsymbol{v}^*)/m} + \text{lower-order terms.} \tag{3}$$

The leading-order error accumulation term $(I-\gamma\boldsymbol{P}^{\pi^*})^{-1}\sqrt{\mathrm{var}(\boldsymbol{v}^*)/m}$ satisfies the so-called *total variance* property, and can be upper bounded uniformly by $\sqrt{(1-\gamma)^{-3}m^{-1}}$, resulting the correct dependence on $(1-\gamma)$. Therefore the monotonicity property allows us to use π^* as a *proxy* policy, which carefully bounds the error accumulation. For the additional subtlety of how to obtain an optimal policy, please refer to Sidford et al. (2018a) for the variance reduction technique and the monotone-policy technique.

Similar observations regarding MDPs was used in Agarwal et al. (2019) as well. This powerful technique, however, does not generalize to the game case due to the *lack of monotonicity*. Indeed, (2) does not hold for stochastic games due to the existence of both minimization and maximization operations in the Bellman operator. This is the critical issue which this paper seeks to overcome.

Finding Monotone Value-Strategy Sequences for Stochastic Games: Analogously to the MDP case, one approach is to bound error accumulation for stochastic games is to bound each entry of the error

vector $\boldsymbol{\epsilon}^{(i)}$ carefully. In fact, our method for solving stochastic games is very much like the MDP method used in Sidford et al. (2018a). However, the analysis is much different in order to resolve the difficulty introduced by the lack of monotonicity.

Since a stochastic game has two players, we modify the variance reduced Q-value iteration (vQVI) method in Sidford et al. (2018a) to obtain a min-player strategy and a max-player strategy respectively. Since the two players are symmetric, let us focus on introducing and analyzing the algorithm for the min-player. By a slight modification of the vQVI method, we can guarantee to obtain a sequence of strategies and values, $\{\boldsymbol{v}^{(i)}, \boldsymbol{Q}^{(i)}, \sigma^{(i)}, \boldsymbol{\epsilon}^{(i)}\}_{i=0}^{R}$, that satisfy, with high probability,

$$\begin{aligned}
&1.\quad \boldsymbol{v}^{(0)} \geq \boldsymbol{v}^{(1)} \geq \ldots \boldsymbol{v}^{(R)} \geq \boldsymbol{v}^{*};\\
&2.\quad \mathcal{T}_{\sigma^{(i)}}[\boldsymbol{v}^{(i)}] \leq \boldsymbol{v}^{(i)}, \mathcal{T}[\boldsymbol{v}^{(i)}] \leq \boldsymbol{v}^{(i)}, \mathcal{H}_{\pi_{\min}^{(i)}}[\boldsymbol{v}^{(i)}] \leq \boldsymbol{v}^{(i)};\\
&3.\quad \boldsymbol{Q}^{(i)} \leq \boldsymbol{Q}[\boldsymbol{v}^{(i-1)}] + \boldsymbol{\epsilon}^{(i)};\\
&4.\quad \boldsymbol{v}^{(i)} \leq V[\boldsymbol{Q}^{(i)}]. \qquad (4)
\end{aligned}$$

where $\sigma^{(i)} = (\pi_{\max}^{(i)}, \pi_{\min}^{(i)})$. The first property guarantees that the value sequences are monotonically decreasing, the second property guarantees $\boldsymbol{v}^{(i)}$ is always an upper bound of the value $\boldsymbol{v}^{\pi_{\min}^{(i)}}$, and the third and fourth inequality guarantees that $\boldsymbol{v}^{(i)}$ is well approximated by $V[\boldsymbol{Q}^{(i)}]$ and the estimation error satisfy $|\boldsymbol{\epsilon}^{(i)}| \lesssim \sqrt{\mathrm{var}(\boldsymbol{v}^{(i)})/m}$, where m is the total number of samples used per state-action pair. Note that, as long as we can guarantee that $\boldsymbol{v}^{(R)} - \boldsymbol{v}^{*} \leq \epsilon$, we can guarantee the min-strategy $\pi_{\min}^{(R)}$ is also good: $\boldsymbol{v}^{*} \leq \boldsymbol{v}^{\pi_{\min}^{(R)}} \leq \boldsymbol{v}^{(R)}$.

Controlling Error Accumulation using Auxiliary Markovian Strategy: Due to the lack of monotonicity (2), we cannot use the optimal strategy σ^{*} as a proxy strategy to carefully account for the error accumulation. To resolve this issue, we construct a new proxy strategy σ^{∞}. This strategy is a Markovian strategy, which is time-dependent but not history dependent, i.e., at time t, the strategy played is a deterministic map $\sigma_t^{\infty} : \mathcal{S} \to \mathcal{A}$. The proxy strategy satisfies the following:

Underestimation. its value, $\boldsymbol{v}[\sigma_i^{\infty}]$, (expected discounted cumulative reward starting from any time) is upper bounded by $\boldsymbol{v}^{*}$;

Contraction.

$$\begin{aligned}
\boldsymbol{v}^{(i)}(s) - \boldsymbol{v}[\sigma_i^{\infty}](s) \leq \gamma \boldsymbol{P}(\cdot|s, \sigma_i^{\infty}(s))^{\top}\left(\boldsymbol{v}^{(i-1)} - \boldsymbol{v}[\sigma_{i-1}^{\infty}]\right)\\
+ \boldsymbol{\epsilon}^{(i)}(s, \sigma_i^{\infty}(s)),
\end{aligned}$$

Similarly, we can bound the error $\boldsymbol{\epsilon}^{(i)}(s, \sigma_i^{\infty}(s))$ by the variance-of-value of the proxy strategy

$$\begin{aligned}
\boldsymbol{\epsilon}^{(i)}(s, \sigma_i^{\infty}(s)) \leq &\sqrt{\mathrm{var}(\boldsymbol{v}[\sigma_i^{\infty}])(s, \sigma_i^{\infty}(s))/m}\\
&+ \text{ lower-order terms.}
\end{aligned}$$

Based on the first property, we can upper bound

$$\boldsymbol{v}^{(i)} - \boldsymbol{v}^{*} \leq \boldsymbol{v}^{(i)} - \boldsymbol{v}[\sigma_i^{\infty}].$$

Based on the second property, and induction on i, we can now write a new form of error accumulation,

$$\begin{aligned}
\boldsymbol{v}^{(R)} - \boldsymbol{v}^{*} \lesssim &\sum_{i=1}^{R} \gamma^{R-i} \boldsymbol{P}^{\sigma_R^{\infty}} \cdot \boldsymbol{P}^{\sigma_{R-1}^{\infty}} \cdot \ldots \cdot \boldsymbol{P}^{\sigma_{i+1}^{\infty}}\\
&\cdot \sqrt{\mathrm{var}(\boldsymbol{v}[\sigma_{i-1}^{\infty}])_{\sigma_i^{\infty}}/m} + \text{ lower-order terms,}
\end{aligned}$$

where $\mathrm{var}(\boldsymbol{v}[\sigma_{i-1}^{\infty}])_{\sigma_i^{\infty}}(s) := \mathrm{var}(\boldsymbol{v}[\sigma_i^{\infty}])(s, \sigma_i^{\infty}(s))$ for all $s \in \mathcal{S}$. We derive a new *law of total variance* bound for the first term and ultimately prove an error accumulation upper bound:

$$\boldsymbol{v}^{(R)} - \boldsymbol{v}^{*} \lesssim \sqrt{(1-\gamma)^{-3}m} + \text{ lower-order terms,}$$

giving the optimal sample bound.

3 Sample Complexity of Stochastic Games

In this section, we provide and analyze our sampling-based algorithm for solving stochastic games. Recall that we have a *generative model* for the game such that we can obtain samples from state-action pairs. Each sample is obtained in time $O(1)$. As such we care about the total number of samples used or the total amount of time consumed by the algorithm. We will provide an efficient algorithm that takes input a generative model and obtains a good strategy for the underlying stochastic game.

We now describe the algorithm. Since the min-player and max-player are symmetric, let us focus on the min-player strategy. For the max player strategy, we can either consider the game $\mathcal{G}' = (\mathcal{S}_{\min}, \mathcal{S}_{\max}, \boldsymbol{P}, \mathbf{1} - \boldsymbol{r}, \gamma)$, in which the roles of the max and min players switched, or use the corresponding algorithm for the max-player defined in Section A.4, an algorithm that is a direct generalization from the min-player algorithm.

The Full Algorithm. For simplicity, let us denote $\beta = 1/(1-\gamma)$. Our full algorithm will use the QVI-MDVSS algorithm (Algorithm 1) as a subroutine. As we will show shortly, this subroutine maintains a monotonic value strategy sequence with high probability. Suppose the algorithm is specified by an accuracy parameter $\epsilon \in (0, 1]$. We initialize a value vector $\boldsymbol{v}^{(0)} = \beta \mathbf{1}$, and an arbitrary strategy $\sigma^{(0)} = (\pi_{\min}^{(0)}, \pi_{\max}^{(0)})$. Let $u^{(0)} = \beta$. Then our initial value and strategy satisfy the requirement of the input specified by Algorithm 1:

$$\begin{aligned}
&\boldsymbol{v}^{*} \leq \boldsymbol{v}^{(0)} \leq \boldsymbol{v}^{*} + u^{(0)}\mathbf{1}, \quad \boldsymbol{v}^{(0)} \geq \mathcal{T}[\boldsymbol{v}^{(0)}],\\
&\text{and} \quad \boldsymbol{v}^{(0)} \geq \mathcal{T}_{\sigma^{(0)}}[\boldsymbol{v}^{(0)}];
\end{aligned}$$

Algorithm 1 QVI-MDVSS: algorithm for computing monotone decreasing value-strategy sequences.

Input: A generative model for stochastic game, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{r}, \boldsymbol{P}, \gamma)$;
Input: Precision parameter $u \in [0, (1-\gamma)^{-1}]$, and error probability $\delta \in (0,1)$;
Input: Initial values $\boldsymbol{v}^{+(0)}, \sigma^{+(0)}$ that satisfies monotonicity:

$$\boldsymbol{v}^* \le \boldsymbol{v}^{+(0)} \le \boldsymbol{v}^* + u\,\mathbf{1}, \quad \boldsymbol{v}^{+(0)} \ge \mathcal{T}[\boldsymbol{v}^{+(0)}], \quad \text{and} \quad \boldsymbol{v}^{+(0)} \ge \mathcal{T}_{\sigma^{+(0)}}[\boldsymbol{v}^{+(0)}]; \tag{5}$$

Output: $\{\boldsymbol{v}^{+(i)}, \boldsymbol{Q}^{+(i)}, \sigma^{+(i)}, \boldsymbol{\xi}^{+(i)}\}_{i=0}^{R}$ which is an MDVSS with probability at least $1-\delta$;

INITIALIZATION:
Let c_1, c_2, c_3, c be some tunable absolute constants;
Initialize constants:
$\beta \leftarrow (1-\gamma)^{-1}$, and $R \leftarrow \lceil c_1 \beta \ln[\beta u^{-1}] \rceil$; $m_1 \leftarrow c_2 \beta^3 \cdot \min(1, u^{-2}) \cdot \log(8|\mathcal{S}||\mathcal{A}|\delta^{-1})$;
$m_2 \leftarrow c_3 \beta^2 \log[2R|\mathcal{S}||\mathcal{A}|\delta^{-1}]$; $\alpha_1 \leftarrow L/m_1$ where $L = c\log(|\mathcal{S}||\mathcal{A}|\delta^{-1}(1-\gamma)^{-1}u^{-1})$;
Obtain an initial batch of samples:
For each $(s,a) \in \mathcal{S} \times \mathcal{A}$: obtain independent samples $s_{s,a}^{(1)}, s_{s,a}^{(2)}, \ldots, s_{s,a}^{(m_1)}$ from $\boldsymbol{P}(\cdot|s,a)$;
Initialize: $\boldsymbol{w}^+ = \widetilde{\boldsymbol{w}}^+ = \widehat{\boldsymbol{\sigma}}^+ = \boldsymbol{Q}^{+(0)} = \boldsymbol{Q}^{+(1)} \leftarrow \beta \cdot \mathbf{1}_{\mathcal{S}\times\mathcal{A}}$ and $i \leftarrow 0$;
for each $(s,a) \in \mathcal{S} \times \mathcal{A}$ **do**
  Compute empirical estimates of $\boldsymbol{P}_{s,a}^\top \boldsymbol{v}^{+(0)}$ and $\mathrm{var}(\boldsymbol{v}^{+(0)})(s,a)$:
  $\widetilde{\boldsymbol{w}}^+(s,a) \leftarrow \frac{1}{m_1}\sum_{j=1}^{m_1} \boldsymbol{v}^{+(0)}(s_{s,a}^{(j)})$; $\widehat{\boldsymbol{\sigma}}^+(s,a) \leftarrow \frac{1}{m_1}\sum_{j=1}^{m_1}(\boldsymbol{v}^{+(0)})^2(s_{s,a}^{(j)}) - (\widetilde{\boldsymbol{w}}^+)^2(s,a)$;
  Shift the empirical estimate to have one-sided error and guarantee monotonicity:
  $\boldsymbol{w}^+(s,a) \leftarrow \widetilde{\boldsymbol{w}}^+(s,a) + \sqrt{\alpha_1 \widehat{\boldsymbol{\sigma}}^+(s,a)} + \alpha_1^{3/4}\beta$
  Compute coarse estimate of the Q-function and make sure its value is in $[0,\beta]$:
  $\boldsymbol{Q}^{+(0)}(s,a) \leftarrow \min[\boldsymbol{r}(s,a) + \gamma \boldsymbol{w}^+(s,a), \beta]$
end for

REPEAT: *successively improve*
for $i = 1$ to R **do**
  Compute the one-step dynamic programming:
  Let $\boldsymbol{v}^{+(i)} \leftarrow \widetilde{\boldsymbol{v}}^{+(i)} \leftarrow \mathcal{T}[\boldsymbol{Q}^{+(i-1)}]$, $\sigma^{+(i)} \leftarrow \widetilde{\sigma}^{+(i)} \leftarrow \sigma(\boldsymbol{Q}^{+(i-1)})$;
  Compute strategy and value and maintain monotonicity:
  For each $s \in \mathcal{S}$ if $\boldsymbol{v}^{+(i)}(s) \ge \boldsymbol{v}^{+(i-1)}(s)$, then $\boldsymbol{v}^{+(i)}(s) \leftarrow \boldsymbol{v}^{+(i-1)}(s)$ and $\sigma^{+(i)}(s) \leftarrow \sigma^{+(i-1)}(s)$;
  Obtaining a small batch of samples:
  For each $(s,a) \in \mathcal{S} \times \mathcal{A}$: draw independent samples $\widetilde{s}_{s,a}^{(1)}, \widetilde{s}_{s,a}^{(2)}, \ldots, \widetilde{s}_{s,a}^{(m_2)}$ from $\boldsymbol{P}(\cdot|s,a)$;
  Compute the expected value, $\boldsymbol{g}^{\pm(i)}$, the estimate of $\boldsymbol{P}[\boldsymbol{v}^{\pm(i)} - \boldsymbol{v}^{\pm(0)}]$ with one-sided error:
  Let $\widetilde{\boldsymbol{g}}^{+(i)}(s,a) \leftarrow \frac{1}{m_2}\sum_{j=1}^{m_2}\left[\boldsymbol{v}^{+(i)}(\widetilde{s}_{s,a}^{(j)}) - \boldsymbol{v}^{+(0)}(\widetilde{s}_{s,a}^{(j)})\right]$;
  Let $\boldsymbol{g}^{+(i)}(s,a) \leftarrow \widetilde{\boldsymbol{g}}^{+(i)}(s,a) + C(1-\gamma)u$, where $C > 0$ is an absolute constant;
  Estimate the approximation error:
  $\boldsymbol{\xi}^{+(i)} \leftarrow 2\sqrt{\alpha_1 \boldsymbol{\sigma}_{\boldsymbol{v}^{+(0)}}} + 2[\alpha_1^{3/4}\beta + C(1-\gamma)u] \cdot \mathbf{1}$
  Improve $\boldsymbol{Q}^{+(i)}$ and make sure its value is in $[0,\beta]$:
  $\boldsymbol{Q}^{+(i+1)} \leftarrow \min\left[\boldsymbol{r} + \gamma \cdot [\boldsymbol{w}^+ + \boldsymbol{g}^{+(i)}], \beta\right]$;
end for
return $\{\boldsymbol{v}^{+(i)}, \boldsymbol{Q}^{+(i)}, \sigma^{+(i)}, \boldsymbol{\xi}^{+(i)}\}_{i=0}^{R}$

Let $u^{(j)} \leftarrow \beta/2^j$ and $\delta \leftarrow 1/\mathrm{poly}(\log(\beta/\epsilon))$. We run Algorithm 1 repeatedly:

$$(v^{(j+1)}, \sigma^{(j+1)}) \leftarrow \text{QVI-MDVSS} \leftarrow (v^{(j)}, \sigma^{(j)}, u^{(j)}, \delta), \tag{6}$$

where $\sigma^{(j)} = (\pi_{\min}^{(j)}, \pi_{\max}^{(j)})$ and we take the terminal value and strategy of the output sequence of Algorithm 1 as the input for the next iteration. In total we run (6) $R' = \Theta(\log(\beta/\epsilon))$ iterations. In the end, we output $\pi_{\min}^{(R')}$ from $\sigma^{(R')} = (\pi_{\min}^{(R')}, \pi_{\max}^{(R')})$ as our min-player strategy.

The formal guarantee of the algorithm is presented in the following theorem.

Theorem 3.1 (Restatement of Theorem 1.1)**.** *Given a stochastic game $\mathcal{G} = (\mathcal{S}_{\min}, \mathcal{S}_{\max}, \boldsymbol{P}, \boldsymbol{r}, \gamma)$ with a generative model, there exists (constructively) an algorithm that outputs, with probability at least $1-\delta$, an ϵ-optimal strategy σ by querying $Z := \widetilde{O}(|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-3}\epsilon^{-2})$ samples in time $O(Z)$ using space $O(|\mathcal{S}||\mathcal{A}|)$ where $\epsilon \in (0,1)$ and $\widetilde{O}(\cdot)$ hides $\mathrm{poly}\log[|\mathcal{S}||\mathcal{A}|/(1-\gamma)/\epsilon/\delta]$ factors.*

The formal proof of Theorem 3.1 is given in the next section. Here we give a sketch of the proof.

Proof Sketch of Theorem 3.1: We first show the

high-level idea. Considering one iteration of (6), we claim that if the input value and strategy $\sigma^{(j)}, \boldsymbol{v}^{(j)}, u^{(j)}$ satisfies the input condition (5), then with probability at least $1-\delta$, the terminal value and strategy of the output sequence, $\sigma^{(j+1)}, \boldsymbol{v}^{(j+1)}$, satisfies,

$$\boldsymbol{v}^{\pi_{\min}^{j+1}} \leq \boldsymbol{v}^{j+1} \leq \boldsymbol{v}^* + u^{(j)}\,\mathbf{1}\,/2 =: \boldsymbol{v}^* + u^{(j+1)}\,\mathbf{1}; \quad (7)$$

and $(\sigma^{(j+1)}, \boldsymbol{v}^{(j+1)}, u^{(j+1)})$ satisfies the the input condition (5). Namely, with high probability, the error of the output is decreased by at least half and the output can be used as an input to the QVI-MDVSS algorithm again. Suppose we run the subroutine of Algorithm 1 for R' times, and conditioning on the event that all the instances of QVI-MDVSS succeed, the final error of $\pi_{\min}^{(R')}$ is then at most $u^{(R')} = 2^{-R'}\beta = \epsilon$, as desired. By setting $\delta = \delta'/R'$ for some $\delta' > 0$, we have that all QVI-MDVSS instances succeed with probability at least $1-\delta'$. It remains to show that the algorithm QVI-MDVSS works as claimed.

High-level Structure of Algorithm 1. To outline the proof, we denote a *monotone decreasing value-strategy sequence* (MDVSS) as $\{\boldsymbol{v}^{(i)}, \boldsymbol{Q}^{(i)}, \sigma^{(i)}, \boldsymbol{\epsilon}^{(i)}\}_{i=0}^{R}$, satisfying (4), where $\boldsymbol{v}^{(i)}, \boldsymbol{\epsilon}^{(i)} \in \mathbb{R}^{\mathcal{S}}, \boldsymbol{Q}^{(i)} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and $\sigma^{(i)} = (\pi_{\min}^{(i)}, \pi_{\max}^{(i)}) \in \mathcal{A}^{\mathcal{S}}$. A more formal treatment of the sequence is presented in Section A.2.

We next introduce the high-level idea of Algorithm 1. The basic step of the algorithm is to do approximate value-iteration while preserving all monotonic properties required by an MDVSS, i.e., we would like to approximate

$$\boldsymbol{Q}^{(i)} = \boldsymbol{Q}[\boldsymbol{v}^{(i-1)}] := \boldsymbol{r} + \boldsymbol{P}\boldsymbol{v}^{(i-1)}$$
$$\text{and} \quad \mathcal{T}[\boldsymbol{v}^{(i-1)}] := V[\boldsymbol{Q}(\boldsymbol{v}^{(i-1)})].$$

We would like to approximate $\boldsymbol{P}\boldsymbol{v}^{(i-1)}$ using samples, but we do not want to use the same amount of samples per iteration (as it become costly if the number of iterations is large). Instead, we compute only the *first* iteration (i.e., estimate $\boldsymbol{P}\boldsymbol{v}^{(0)}$) up to high accuracy with a large number of samples (m_1 samples, defined in Line 9). These computations are presented in Line 15-20. To maintain an upper bound of the of the estimation error, we also compute the empirical variances of the updates in Line 16. We shift upwards our estimates by the estimation error upper bounds to make our estimators one-sided, which is crucial to maintain the MDVSS properties. For the subsequent steps (Line 26 - 37), we use m_2 samples per iteration ($m_2 \ll m_1$) to estimate $\boldsymbol{P}(\boldsymbol{v}^{(i)} - \boldsymbol{v}^{(0)})$. The expectation is that $(\boldsymbol{v}^{(i)} - \boldsymbol{v}^{(0)})$ has a small ℓ_∞ norm, and hence $\boldsymbol{P}(\boldsymbol{v}^{(i)} - \boldsymbol{v}^{(0)})$ can be estimated up to high accuracy with only a small number of samples. The estimator of $\boldsymbol{P}(\boldsymbol{v}^{(i)} - \boldsymbol{v}^{(0)})$ plus the estimator of $\boldsymbol{P}\boldsymbol{v}^{(0)}$ in the initialization steps gives a high-accuracy estimator (Line 37) for the value iteration. Since $m_2 \ll m_1$, the total number of samples per state-action pair is dominated by m_1. This idea is formally known as *variance-reduction*, firstly proposed for solving MDP in Sidford et al. (2018b). Similarly, we shift our estimators to be one-sided. We additionally maintain carefully-designed strategies in Line 26-28 to preserve monotonicity. Hence the algorithm can be viewed as a value-strategy iteration algorithm.

Correctness of Algorithm 1. We now sketch the proof of correctness for Algorithm 1. Firstly Proposition (A.3) shows that the if an MDVSS, e.g., $\{\boldsymbol{v}^{+(i)}, \boldsymbol{Q}^{+(i)}, \sigma^{+(i)}, \boldsymbol{\epsilon}^{+(i)}\}_{i=0}^{R}$, satisfies $\|\boldsymbol{v}^{+(R)} - \boldsymbol{v}^*\|_\infty \leq \epsilon$ for some $\epsilon > 0$ then their terminal strategies and values satisfy

$$\boldsymbol{v}^{\pi_{\min}^{+(R)}} \leq \boldsymbol{v}^{+(R)} \leq \boldsymbol{v}^* + \epsilon\,\mathbf{1}\,.$$

This indicates that as long as we can show $\epsilon \leq u/2$, then the *halving-error-property* (7) holds.

Proposition A.4 shows the halving-error-property can be achieved by setting

$$\boldsymbol{\epsilon}^{+(i)} \lesssim \sqrt{\mathrm{var}(\boldsymbol{v}^{+(0)})/m} + \text{ lower-order terms},$$

where $\mathrm{var}(\boldsymbol{v}^{+(0)})$ is the variance-of-value vector of $\boldsymbol{v}^{+(0)}$ and $m \gtrsim \sqrt{\beta^3 u^{-2}}$. This proof is based on constructing an auxiliary Markovian strategy for analyzing the error accumulation throughout the value-strategy iterations. The Markovian strategy is a time-dependent strategy used as a proxy for analyzing the entrywise error recursion (Lemmas A.4-A.11).

Proposition A.12 shows, with high probability, Algorithm 1 produces value-strategy sequences $\{\boldsymbol{v}^{+(i)}, \boldsymbol{Q}^{+(i)}, \sigma^{+(i)}, \boldsymbol{\xi}^{+(i)}\}_{i=0}^{R}$, which is indeed an MDVSS and $\boldsymbol{\xi}^{+(i)}$ satisfies Proposition A.4. The proof involves analyzing the probability of "good events" on which monotonicity is preserved at every iteration by using confidence estimates computed during the iterations and concentration arguments. See Lemmas A.13-A.18 for the full proof of Proposition A.12.

Putting Everything Together. Finally by putting together the strategies, we conclude that the terminal strategy of the iteration (6) is always an approximately optimal min-player strategy to the game, with high probability. For implementation, since our algorithm only computes the inner product based on samples, the total computation time is proportional to the number of samples. Moreover, since we can update as samples are drawn and output the monotone sequences as they are generated, we do not need to store samples or the value-strategy sequences, thus the overall space is $O(|\mathcal{S}||\mathcal{A}|)$. □

References

Agarwal, A., Kakade, S., and Yang, L. F. (2019). On the optimality of sparse model-based planning for markov decision processes. *arXiv preprint arXiv:1906.03804*.

Arslan, G. and Yüksel, S. (2017). Decentralized q-learning for stochastic teams and games. *IEEE Transactions on Automatic Control*, 62(4):1545–1558.

Azar, M. G., Munos, R., Ghavamzadeh, M., and Kappen, H. (2011a). Reinforcement learning with a near optimal rate of convergence.

Azar, M. G., Munos, R., Ghavamzadeh, M., and Kappen, H. (2011b). Speedy q-learning. In *Advances in neural information processing systems*.

Azar, M. G., Munos, R., and Kappen, H. J. (2013). Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bowling, M. and Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, volume 17, pages 1021–1026. Lawrence Erlbaum Associates Ltd.

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

De Ghellinck, G. (1960). Les problemes de decisions sequentielles. *Cahiers du Centre dEtudes de Recherche Opérationnelle*, 2(2):161–179.

d'Epenoux, F. (1963). A probabilistic production and inventory problem. *Management Science*, 10(1):98–108.

Even-Dar, E., Mannor, S., and Mansour, Y. (2006). Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105.

Filar, J. and Vrieze, K. (2012). *Competitive Markov decision processes*. Springer Science & Business Media.

Fudenberg, D. and Tirole, J. (1991). *Game theory*. MIT Press, Cambridge, MA.

Hansen, T. D., Miltersen, P. B., and Zwick, U. (2013). Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM)*, 60(1):1.

Howard, R. A. (1960). *Dynamic programming and Markov processes*. The MIT press, Cambridge, MA.

Hu, J. and Wellman, M. P. (1999). Multiagent reinforcement learning in stochastic games. *Submitted for publication*.

Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069.

Hu, J., Wellman, M. P., et al. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250. Citeseer.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England.

Kearns, M. J. and Singh, S. P. (1999). Finite-sample convergence rates for q-learning and indirect algorithms. In *Advances in neural information processing systems*, pages 996–1002.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on International Conference on Machine Learning*, ICML'94, pages 157–163, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Littman, M. L. (1996). Algorithms for sequential decision making.

Littman, M. L. (2001a). Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pages 322–328.

Littman, M. L. (2001b). Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55–66.

Mansour, Y. and Singh, S. (1999). On the complexity of policy iteration. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 401–408. Morgan Kaufmann Publishers Inc.

Maskin, E. and Tirole, J. (2001). Markov perfect equilibrium. I. Observable actions. *J. Econom. Theory*, 100(2):191–219.

Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pages 286–295.

Perolat, J., Scherrer, B., Piot, B., and Pietquin, O. (2015). Approximate dynamic programming for two-player zero-sum markov games. In *International Conference on Machine Learning (ICML 2015)*.

Pérolat, J., Scherrer, B., Piot, B., and Pietquin, O. (2015). Approximate dynamic programming for two-player zero-sum markov games. In *Proceedings of the 32th International Conference on International Conference on Machine Learning*, ICML'15.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Scherrer, B. (2013). Improved and generalized upper bounds on the complexity of policy iteration. In *Advances in Neural Information Processing Systems*, pages 386–394.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100.

Sidford, A., Wang, M., Wu, X., Yang, L., and Ye, Y. (2018a). Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems*, pages 5186–5196.

Sidford, A., Wang, M., Wu, X., and Ye, Y. (2018b). Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–787. Society for Industrial and Applied Mathematics.

Singh, S. P. and Yee, R. C. (1994). An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233.

Wang, M. (2017). Randomized linear programming solves the discounted Markov decision problem in nearly-linear running time. *arXiv preprint arXiv:1704.01869*.

Wei, C.-Y., Hong, Y.-T., and Lu, C.-J. (2017). Online reinforcement learning in stochastic games. In *Advances in Neural Information Processing Systems*, pages 4994–5004.

Ye, Y. (2005). A new complexity result on solving the Markov decision problem. *Mathematics of Operations Research*, 30(3):733–749.

Ye, Y. (2011). The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603.