

The following section proves complexity of offline learning in MDPs and presents an algorithm. Following that we extend the results and the algorithm to zero-sum and common-payoff SGs and discuss the complications that arise due to the opponent.

2 Background

Although the concepts in this section are likely well known by the reader, since part of our message is that there is much to be gained by being explicit about the setting, we want to err on the side of explicitness.

2.1 Markov Decision Processes

Definition 2.1 A Markov Decision Process (MDP) is a tuple $M = (S, A, p, r)$, where

- S is a finite set of states of the world. $|S| = n$.
- A is a finite set of actions. $|A| = m$.
- $p : S \times A \rightarrow \Delta(S)$ is the probability transition function, giving for each state and action a probability distribution over next states.
- $r : S \times A \rightarrow \mathfrak{R}$ is the reward function, giving the expected immediate reward.

Without loss of generality we will assume that the rewards lie in the $[-r_{max}/2, r_{max}/2]$ range.

Definition 2.2 An MDP with known reward function and state transition probabilities is called a known MDP. Otherwise it is called an unknown MDP.

A policy $\pi : S \rightarrow A$ prescribes which action to take in every state of the MDP. Associated with every policy π is a value $V^\pi(t) : S \rightarrow \mathfrak{R}$ which is the accumulated reward, either discounted or undiscounted, from following this policy for t steps. As $t \rightarrow \infty$, the value of t step policy approaches the value V^π of infinite-horizon policy. V^π can be computed by solving a system of linear equations. In this paper we are interested in discounted and undiscounted¹ (average return) problems, so we will use two definitions of V^π :

Definition 2.3 (Discounted)

$$V^\pi(s) = r(s, \pi(s)) + \alpha \sum_{s'} p(s, \pi(s), s') V^\pi(s').$$

Definition 2.4 (Undiscounted)

$$V^\pi(s) + h^\pi(s) = r(s, \pi(s)) + \sum_{s'} p(s, \pi(s), s') h^\pi(s').$$

Notice that because MDP is unichain, for undiscounted reward $V^\pi(s) = V^\pi(s'), \forall s, s'$.

We are interested in computing the optimal policy:

$$\pi^* = \operatorname{argmax}_\pi V^\pi$$

¹When speaking about undiscounted MDPs, we assume they are unichain [Bertsekas & Tsitsiklis 1996, Section 4.2].

2.2 Stochastic Games

Stochastic Games can be seen as either an extension of a normal form game to multiple states or as an extension of MDP to multi player setting. Both of these views will be useful to us. We confine ourselves to 2-player games.

Definition 2.5 Stochastic Game (SG) is a tuple $G = (S, A_1, A_2, p, r_1, r_2)$, where

- S is a finite set of states of the world. $|S| = n$.
- A_1 is a finite set of player's actions. $|A_1| = m_1$.
- A_2 is a finite set of opponent's actions. $|A_2| = m_2$.
- $p : S \times A_1 \times A_2 \rightarrow \Delta(S)$ is the probability transition function, giving for each state, player's actions and opponent's actions, a probability distribution over next states.
- $r_i : S \times A_1 \times A_2 \rightarrow \mathfrak{R}$ is the reward function, giving the expected immediate reward for player i .

Without loss of generality we will assume that the rewards lie in the $[-r_{max}/2, r_{max}/2]$ range. Also let $m = m_1 * m_2$.

The players have policies $\pi : S \rightarrow A_1$ and $\phi : S \rightarrow A_2$. A t -step value of a policy pair (π, ϕ) is $V^{\pi, \phi}(t) : S \rightarrow \mathfrak{R}$. The value of infinite-horizon policy is denoted by $V^{(\pi, \phi)}$. In a SG, every state s together with the sets of actions available to both players can be thought of as a normal form (or static) game g . We will work with zero-sum stochastic games (ZS-SG), where each static game is zero-sum (the players rewards sum to zero²) under discounted (DZS-SG) and undiscounted³ (AZS-SG) reward formulations. Both DZS-SG and AZS-SG have an optimal value and there exist optimal stationary policies [Filar & Vrieze 1997]. We will also examine common-payoff (collaborative or cooperative) stochastic games (CP-SG) where each static game is common-payoff (the players' reward functions are equal) under discounted (DCP-SG) and undiscounted (ACP-SG) reward formulations. We define optimal value and optimal stationary policies in DCP-SG and ACP-SG by assuming that the players are able to coordinate in each stage game. Then these games can be solved as MDPs. We discuss our assumption further in Section 5.

3 Single Player Reinforcement Learning

In undiscounted setting traditional algorithms such as E^3 and $R - max$ have concentrated on the online problem. It is important to realize that this problem has two sources of difficulty - one having to do with the speed of learning the unknown parameters, and the other having to do with the speed of accumulating rewards. Unlike the former, the latter has nothing to do with the unknown nature of the problem - it is manifested already in fully known MDPs. The key insight of these algorithms is the identification of the mixing time.

Definition 3.1 ([Kearns & Singh 1998]) The ϵ -return mixing time T_ϵ of policy π in undiscounted MDP M is the smallest T such that for all $t \geq T$, $|V^\pi(t, i) - V^\pi(i)| \leq \epsilon$ for all states i .

²In fact, we can generalize slightly to constant-sum games.

³When speaking about undiscounted SGs, we assume they are irreducible [Filar & Vrieze 1997, Section 5.1].

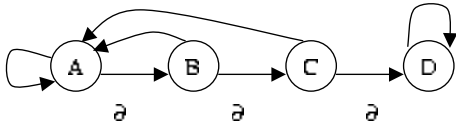


Figure 1: Exponential Mixing Time

In known MDPs, given a known optimal policy, the complexity of accumulating near-optimal rewards is linear in T_ϵ . The T_ϵ depends on the number of states, the connectivity of the graph and on the probability transition function. In general it can be exponential in the number of states. To see that, look at the Markov Chain in Figure 1. If the reward at state D is larger than at other states, the optimal return will be achieved after the system moves to state D . But starting from state A the number of transitions necessary to get to state D is proportional to $(1/\delta)^3$.

For discounted setting, a PAC-based algorithm addresses both offline [Fiechter 1994] and online [Fiechter 1997] problems. However, the algorithm learns good policies from a fixed initial state which is a reasonable assumption for on-line problem, but not for offline one. In addition, offline problem requires the presence of “reset” button.

In this paper we concentrate on the offline problem, making no assumptions on the initial state or special sampling procedures. In undiscounted setting we could use $E3$ or $R - max$ to compute a near-optimal policy, but does the mixing time give a satisfactory bound? Not always. To see that, consider any MDP with an exponential T_ϵ . Now add a new state which has actions leading to every state and add a new action to all states that leads to the new state. If the reward on going to and from the new state is very low, then T_ϵ will not change significantly. On the other hand, sampling the whole state space can be done very fast, since we basically have a reset button available. And having the whole model, we should be able to compute a near-optimal policy fast. In general, we can bound the sampling complexity by the spectral radius.

Definition 3.2 Given an MDP, let $o(s, \pi, s')$ be some path (s, i_1, \dots, i_l, s') from state s to state s' under policy π . Let $P(s, \pi, s') = \max_{o(s, \pi, s')} \prod_k p(i_k, \pi(i_k), i_{k+1})$. Then for any MDP we define its spectral radius r_s as $1/[\min_{s, s'} \max_{\pi} P(s, \pi, s')]$. Similarly, for any SG we define its spectral radius r_s as $1/[\min_{s, s', \varphi} \max_{\pi} P(s, \pi, \varphi, s')]$.

The complexity of sampling the whole state-space is polynomial in the spectral radius. The spectral radius depends on the number of states, the connectivity of the graph and on the probability transition function. In general it can be exponential in the number of states. However, it is not comparable to ϵ -return mixing time - it can be exponentially less or greater. We saw above the case when it is exponentially less. It is exponentially greater (see Figure 2) whenever there is a set of states with exponential sampling time (states A, B, C, D), whose low reward excludes them from the optimal policy (go directly to state E) and hence from affecting T_ϵ .

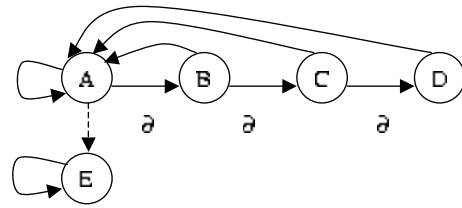


Figure 2: Exponential Spectral Radius

We will be interested in a complexity bound on learning a near-optimal policy that uses the minimum of T_ϵ and r_s for the undiscounted setting and uses r_s for the discounted setting. Our approach consists of three steps. First, we note that in known MDPs, the computation of an optimal policy - unlike the accumulation of optimal reward - can be done in fast.

Theorem 3.1 (MDP Complexity) *The optimal policy in discounted and undiscounted setting for known MDPs can be computed using linear programming in $\text{poly}(n, m)$ time [Bertsekas 1995].*

(Technically speaking, there is an extra term B , the maximum number of bits necessary to represent any parameter of MDP, in the complexity equation. But it is present in all our formulas and does not add new insight, so we omit it.)

Then we note that the optimal policy is continuous in the parameters of the MDP, i.e. rewards and probability transitions. To show that we first define a perturbed MDP.

Definition 3.3 Let M be an MDP, and M_ϵ be an ϵ -perturbed MDP with the values of all its parameters differing from M by less than ϵ . Let π be some policy and V^π be its value (either discounted or undiscounted) in M , and V_ϵ^π its value (discounted or undiscounted respectively) in M_ϵ . Let π^* be the optimal policy in M and π_ϵ^* be the optimal policy in M_ϵ .

Theorem 3.2 (MDP Continuity I) *The value of policy π is continuous in some neighborhood $\xi > 0$ w.r.t. perturbations in parameter space of the MDP for discounted and undiscounted settings. That is,*

$$|V^\pi - V_\epsilon^\pi| < C\epsilon, \text{ where } C \in \mathbb{R}^+, \epsilon < \xi.$$

Proof: Follows from the fact that MDPs can be solved using linear programming, the solution of which is continuous in the inputs. For discounted MDPs, $\xi = \infty$. For undiscounted MDPs, ξ must be small enough to preserve unichain property of the MDP. It is easy to see that $\xi \geq \min_{s, a, s'} p(s, a, s')$. ■

Corollary 3.3 (MDP Continuity II) *The optimal strategy in a perturbed game is nearly optimal in the original game. That is,*

$$|V^{\pi^*} - V^{\pi_\epsilon^*}| < C\epsilon, \text{ where } C \in \mathbb{R}^+, \epsilon < \xi.$$

From this it follows that if we can sample the MDP in polynomial time, we can compute a near-optimal policy in polynomial time. First we need to determine how much we need to sample each unknown parameter. We estimate that using variants of the Chernoff bound.

Theorem 3.4 (Hoeffding) Given a random variable r bounded by $[-r_{max}/2, r_{max}/2]$ and a set of its sampled values r_1, \dots, r_k , for any $\epsilon > 0$,

$$P \left\{ \left| \frac{1}{k} \sum r_i - Er \right| \geq \epsilon \right\} \leq 2e^{-k\epsilon^2/r_{max}^2},$$

where Er is the expectation of r .

So the number of samples necessary for given levels of ϵ and δ is polynomial and given by

$$k(\epsilon, \delta) = \frac{r_{max}^2}{\epsilon^2} \left(-\ln \frac{\delta}{2} \right).$$

Similarly for probability transition functions.

Theorem 3.5 (Glivenko-Cantelli) If $F(z)$ is the true pdf and $F_k(z)$ is the standard empirical pdf, then for any $\epsilon > 0$,

$$P \{ \sup_{z \in \mathbb{R}} |F(z) - F_k(z)| \geq \epsilon \} \leq 8ke^{-k\epsilon^2/32}.$$

We are now ready to state the main result of this section.

Algorithm 3.1 The algorithm builds two MDPs, \hat{M} and M_ϵ . Initially, both are empty. After each sample newly discovered states and actions are added to both MDPs. The probability transitions are also updated as standard empirical pdf. The MDPs differ in their updates of rewards. The M_ϵ is build using sampled rewards, while the \hat{M} is assigned fictional rewards that facilitate exploration. In particular, given $\epsilon, \delta, r_{max}$ that specify model accuracy, the number of parameter samples required is computed via Theorems 3.4 and 3.5. Then, for MDP \hat{M} , $r(s, a)$ is set equal to the number of samples required minus the number of times that reward was actually sampled. The minimum value for $r(s, a)$ is 0.

1. Initialize T_s to some value for all states s .
2. From state s compute optimal T -step policy $\pi_T^*(s)$ in the MDP \hat{M} .
3. If reward from $\pi_T^*(s)$ is 0 then increase T_s .
4. Follow $\pi_T^*(s)$ for T steps, updating \hat{M} and M_ϵ .
5. Goto step 2.

We compare this algorithm to existing approaches. It explores strictly more than E^3 and $R - max$ in undiscounted MDPs, hence finding ϵ -optimal policy in $poly(n, m, 1/\epsilon, 1/\delta, T_\epsilon)$ time. It is also easy to see that after $poly(n, m, 1/\epsilon, 1/\delta, r_s)$ steps it will visit the whole state space and M_ϵ can be used to compute ϵ -optimal policy. Finally, for discounted MDPs, this result extends PAC-learning [Fiechter 1994] by removing dependence on the discount rate.

Theorem 3.6 (Main) Using Algorithm 3.1 the ϵ -optimal policy for an unknown MDP can be computed with probability $1 - \delta$ in $poly(n, m, 1/\epsilon, 1/\delta, \min(r_s, T_\epsilon))$ for the undiscounted setting and in $poly(n, m, 1/\epsilon, 1/\delta, r_s)$ for the discounted setting.

4 Reinforcement Learning in Zero-Sum Stochastic Games

The discussion of learning in stochastic games cannot be removed from the discussion of algorithms that react to the opponent behavior. In general SGs, even the notion of an optimal policy is suspect for a non-fixed opponent. However, in some classes of SGs we have a good definition of optimality, and for these classes we are able to extend the framework from the previous section. In particular, we cover zero-sum and common-payoff SGs. We start with the former in this section.

We are not aware of any existing results on offline reinforcement learning in DZS-SGs. For AZS-SGs we could use $R - max$, but as before we can improve upon the complexity bound by using spectral radius together with the mixing time. The first steps are the same as in the previous section. We start by showing that computing a near-optimal policy in a known ZS-SG is fast.

Theorem 4.1 (ZS-SG Complexity) Value Iteration in ZS-SGs converges to an ϵ -optimal policy exponentially fast. For DZS-SG the convergence rate depends on n, m, α , while for AZS-SG it depends on n, m, r_s .

Proof: See Chapters 11 and 13 of [Wal 1981] for convergence rates of value iteration in DZS-SG and AZS-SG respectively. We should note that for fixed α and r_s the ϵ -optimal policy in DZS-SG and AZS-SG respectively can be computed in time polynomial in these quantities. We also believe it is possible to remove the dependence on the α and r_s by using nonlinear programming, instead of value iteration (see, for example, Chapter 3 in [Filar & Vrieze 1997]). ■

Next we note that the optimal policy is continuous in the parameters of the ZS-SG. To show that we first define a perturbed ZS-SG.

Definition 4.1 Let G be a ZS-SG. Then G_ϵ is called ϵ -perturbed ZS-SG if the values of all its parameters differ from G by less than ϵ . We use subscript ϵ when referring to quantities of G_ϵ , for example $V_\epsilon^{(\pi^*, \phi^*)}$ or π_ϵ .

Theorem 4.2 (ZS-SG Continuity I) The optimal value $V^{(\pi^*, \phi^*)}$ of DZS-SG or AZS-SG is continuous in some neighborhood $\xi > 0$ w.r.t. perturbations in parameter space of that SG. That is,

$$\left| V^{(\pi^*, \phi^*)} - V_\epsilon^{(\pi^*, \phi^*)} \right| < C\epsilon, \text{ where } C \in \mathbb{R}^+, \epsilon < \xi.$$

Proof: See [Filar & Vrieze 1997], Theorem 4.3.7 for proof in DZS-SG case, with $\xi = \infty$. For AZS-SG we assume that parameter perturbations within some radius ξ do not change game irreducibility. It is easy to see from the definition of irreducible matrix that $\xi \geq \min_{s, a_1, a_2, s'} p(s, a_1, a_2, s')$. Let Θ denote all parameters of the game. Then we can write the value function explicitly as $V^{(\pi, \phi)}(\Theta)$. Now, we know that $V^{(\pi, \phi)}(\Theta)$ is continuous in Θ because it can be computed by solving a system of linear equations (Bellman's equations). Moreover, $V^{(\pi, \phi^*)}(\Theta)$ is also continuous in Θ since for a fixed π this game becomes an MDP. Finally, $V^{(\pi^*, \phi^*)}(\Theta) =$

$\max_{\pi} V^{(\pi, \phi^*)}(\Theta)$ is continuous in Θ as well because it is a composition of two continuous functions. ■

Corollary 4.3 (ZS-SG Continuity II) *Optimal strategy π_{ϵ}^* in a perturbed game is nearly optimal in the original game. That is,*

$$\left| V^{(\pi^*, \phi^*)} - V^{(\pi_{\epsilon}^*, \phi^*)} \right| < C\epsilon, \text{ where } C \in \mathbb{R}^+, \epsilon < \xi.$$

Proof: Neighborhood ξ is defined as in Theorem 4.2. See (Theorem 4.3.10, [Filar & Vrieze 1997]) for proof in DZS-SG case. For AZS-SG we have from Theorem 4.2

$$\begin{aligned} \left| V_{\epsilon}^{(\pi_{\epsilon}^*, \phi^*)} - V^{(\pi_{\epsilon}^*, \phi^*)} \right| &< C\epsilon \\ \left| V^{(\pi^*, \phi^*)} - V_{\epsilon}^{(\pi_{\epsilon}^*, \phi^*)} \right| &< C\epsilon \end{aligned}$$

Therefore,

$$\begin{aligned} \left| V^{(\pi^*, \phi^*)} - V^{(\pi_{\epsilon}^*, \phi^*)} \right| &< \\ \left| V^{(\pi^*, \phi^*)} - V_{\epsilon}^{(\pi_{\epsilon}^*, \phi^*)} + V_{\epsilon}^{(\pi_{\epsilon}^*, \phi^*)} - V^{(\pi_{\epsilon}^*, \phi^*)} \right| &< \\ \left| V^{(\pi^*, \phi^*)} - V_{\epsilon}^{(\pi_{\epsilon}^*, \phi^*)} \right| + \left| V_{\epsilon}^{(\pi_{\epsilon}^*, \phi^*)} - V^{(\pi_{\epsilon}^*, \phi^*)} \right| &< 2C\epsilon \end{aligned}$$

■ From this it follows that if we can sample the ZS-SG in polynomial time, we can compute a near-optimal policy in polynomial time. We know how much to sample each unknown parameter from Theorems 3.4 and 3.5. However, we run into a complication - the opponent can prevent the player from ever exploring some actions. We overcome this difficulty by showing that at any point in time the player always has a policy that is either better than the minimax or leads to fast sampling of unknown actions. In effect, by hiding information the opponent can either hurt himself or at best prolong the discovery of minimax policy for polynomial time. This is a special property of zero-sum games. First, we introduce subgames.

Definition 4.2 *A subgame \hat{G} of stochastic game G is identical to G except that the opponent is limited to playing a nonempty subset of his actions in every state.*

Any SG is trivially a subgame of itself. So is any MDP formed by fixing opponent's policy. Such subgames are asymmetric for agents and there is an exponential number of them. Any subgame of a zero-sum SG is zero-sum. We note that by restricting his actions to a subgame, the opponent can only hurt himself.

Theorem 4.4 *The value $V^{(\hat{\pi}^*, \hat{\varphi}^*)}$ of the subgame \hat{G} is at least as large as the value $V^{(\pi^*, \varphi^*)}$ of the ZS-SG G .*

Proof: Denote by Π and Φ policy spaces in G and by $\hat{\Pi}$ and $\hat{\Phi}$ policy spaces in \hat{G} . Then,

$$\begin{aligned} V^{(\hat{\pi}^*, \hat{\varphi}^*)} &= \min_{\varphi \in \hat{\Phi}} \max_{\pi \in \hat{\Pi}} V^{(\pi, \varphi)} \geq \min_{\varphi \in \Phi} V^{(\pi^*, \varphi)} \geq \\ &\geq \min_{\varphi \in \Phi} V^{(\pi^*, \varphi)} = V^{(\pi^*, \varphi^*)} \end{aligned}$$

■ Now we can show that in any partially known game, as long as the player knows all the rewards for one opponent action, the player always has a good strategy.

Corollary 4.5 *In a partially known ZS-SG the player always has a policy such that if the opponent plays only known actions, the player's reward will be at least minimax. If the partially known ZS-SG is actually a subgame, then the player's policy is best response versus that subgame.*

We omit the full presentation of the algorithm due to space constraints, but present an informal overview here. The algorithm has explicit, but alternating stages. The agent starts exploring, quickly finds ϵ -best response (guaranteed to be as good as ϵ -optimal) to the opponent's strategy by exploring all the actions that opponent plays, and starts exploiting. Then, once the opponent changes his strategy to include some previously unknown action, the agent immediately notices that, and once these unknown actions become known, he changes his strategy. The opponent can include previously unknown actions in his strategy only a polynomial number of times due to finite state and action spaces. The opponent can however prolong these changes indefinitely, therefore there is no finite time after which the algorithm can stop learning, unless the whole state-space has been explored. But we can guarantee that after each switch to unknown actions by the opponent, the agent learns ϵ -best response in polynomial time.

We compare this algorithm to existing approaches. It explores strictly more than $R - \max$ in DZS-SGs, hence finding ϵ -best response policy in $\text{poly}(n, m, 1/\epsilon, 1/\delta, T_{\epsilon})$ time. Also, for any opponent play, the agent finds this policy in $\text{poly}(n, m, 1/\epsilon, 1/\delta, r_s)$ time by exploring the whole state-space.

Theorem 4.6 (Main) *For any opponent play in ZS-SGs the agent can compute ϵ -best response, guaranteed to be at least as good as ϵ -optimal, with probability $1 - \delta$ in time $\text{poly}(n, m, 1/\epsilon, 1/\delta, \min(T_{\epsilon}, r_s))$ for AZS-SG and in time $\text{poly}(n, m, 1/\epsilon, 1/\delta, r_s, 1/\ln \alpha)$ for DZS-SG.*

Our results are similar in complexity to $R - \max$ for AZS-SGs. The results on complexity of DZS-SGs are new to the best of our knowledge.

5 Reinforcement Learning in Common-Payoff Stochastic Games

In CP-SGs, assuming rational agents, optimal value is well defined together with a set of optimal policies. Rational agents try to coordinate in every stage game because that is in their best interest. We extend the framework from Section 3 for offline reinforcement learning. The first steps are the same. We note that assuming rational agents, optimal policies for a known CP-SG can be computed fast.

Theorem 5.1 (CP-SG Complexity) *The set of all optimal policy pairs for a CP-SG under discounted and undiscounted formulations can be computed in time $\text{poly}(n, m)$.*

Proof: Since the opponent will maximize his and therefore the agents' payoffs, it is sufficient to consider an MDP version of CP-SG formed by using the same state space S , same action space A_1 , but with payoffs resulting from coordinated opponent behavior, $r(s, a) = \max_b r(s, a, b)$, and probabilities corresponding to that action choice, $p(s, a, s') =$

$p(s, a, b_{(s,a)}^*, s')$, where $b_{(s,a)}^* = \arg \max_b r(s, a, b)$. The optimal policies in this MDP can be computed using linear programming. ■

Next we note that the optimal policy is continuous in the parameters of the CP-SG. The perturbed CP-SG is defined similarly to Definition 4.1.

Theorem 5.2 (CP-SG Continuity I) *The optimal value $V(\pi^*, \varphi^*)$ of DCP-SG or ACP-SG is continuous in some neighborhood $\xi > 0$ w.r.t. perturbations in parameter space of that SG. That is,*

$$\left| V(\pi^*, \varphi^*) - V_\epsilon(\pi^*, \varphi^*) \right| < C\epsilon, \text{ where } C \in \mathbb{R}^+, \epsilon < \xi.$$

Corollary 5.3 (CP-SG Continuity II) *Optimal strategy in a perturbed game is nearly optimal in the original game. That is,*

$$\left| V(\pi^*, \varphi^*) - V(\pi_\epsilon^*, \varphi^*) \right| < C\epsilon, \text{ where } C \in \mathbb{R}^+, \epsilon < \xi.$$

From this it follows that if we can sample the CP-SG in polynomial time, we can compute a near-optimal policy in polynomial time. We know how much to sample each unknown parameter from Theorems 3.4 and 3.5. Unlike zero-sum games, in common-payoff games the agents want to explore the whole state-space and do not hide parts of it from each other. There is only one difficulty - coordinating on the same optimal policy. If the agents do not coordinate, their actions can mismatch resulting in suboptimal rewards. If the agents have access to some coordination device (i.e. communications or coin flips) or the game has focal points, then the problem becomes simple. We will not use any such scheme. Instead we rely on action randomization and a suitable algorithm to make sure that the agents settle on the same optimum. We believe that because agents have common interest, it is reasonable to assume that they will use the same coordination algorithm.

Algorithm 5.1 *The algorithm builds two CP-SGs, \hat{G} and G_ϵ , that are updated similarly to the MDPs in Algorithm 3.1. The value of a state is taken as the coordination optimum of the stage game.*

1. Initialize T_s to some value for all states s .
2. From state s compute optimal T -step policy $\pi_T^*(s)$ in the CP-SG \hat{G} . This policy is computed using dynamic programming.
3. If reward from $\pi_T^*(s)$ is 0 then increase T_s .
4. Follow $\pi_T^*(s)$ for T steps, updating \hat{G} and G_ϵ .
5. Goto step 2.

At the end of the algorithm, once all near-optimal policies are computed, the agents enter into a coordination phase. They start randomizing among their optimal actions (derived from optimal policies) in each state. If in some state they have by chance played an optimal action pair (and they can detect this), then they settle on these actions in that state. If in a state with “settled” actions the opponent chooses some other action, the agent starts randomizing among optimal actions again (this could happen if the opponent is still exploring).

Theorem 5.4 (Main) *The agents can coordinate choosing ϵ -optimal policy pair in unknown CP-SGs with probability $1 - \delta$ in $\text{poly}(n, m, 1/\epsilon, 1/\delta, r_s)$ time for discounted setting and in $\text{poly}(n, m, 1/\epsilon, 1/\delta, \min(T_\epsilon, r_s))$ time for undiscounted setting.*

Proof: The proof is based on the same result for MDPs and on the fact that during coordination stage the probability of not playing any optimal action pair in l trials for any state decreases as $\exp(-l)$. ■

All of the results in this section are new to the best of our knowledge.

6 Summary

In this paper we accent the distinction between online and offline reinforcement learning and present a simple, uniform framework for proving polynomial complexity in MDPs, zero-sum and common-payoff SGs. As in recent work our emphasis has been on the algorithms with provable properties, but we believe in the long run they will be useful for constructing practical algorithms. In the future paper we hope to show that our framework also provides a unifying simple view of the online problem.

7 Acknowledgements

This work was supported in part by DARPA grant F30602-00-2-0598. The authors would also like to thank members of Stanford University Multi-Agent Research Group.

References

- Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific.
- Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control, Vols. I and II*. Belmont, MA: Athena Scientific.
- Brafman, R., and Tennenholtz, M. 2001. R-max: A general polynomial time algorithm for Near-Optimal reinforcement learning. In *Proc. of the 17th International Conf. on Artificial Intelligence (IJCAI-01)*, 953–958. San Francisco, CA: Morgan Kaufmann Publishers, Inc.
- Fiechter, C.-N. 1994. Efficient reinforcement learning. In *Proc. of the 7th Annual ACM Conf. on Computational Learning Theory*, 88–97. New Brunswick, New Jersey: ACM Press.
- Fiechter, C.-N. 1997. Expected mistake bound model for on-line reinforcement learning. In *Proc. of the 14th International Conf. on Machine Learning*, 116–124.
- Filar, J., and Vrieze, K. 1997. *Competitive Markov Decision Processes*. New York, NY: Springer-Verlag.
- Kearns, M., and Singh, S. 1998. Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conf. on Machine Learning*, 260–268. Morgan Kaufmann, San Francisco, CA.
- Wal, J. V. D. 1981. *Stochastic Dynamic Programming*, volume 139 of *Mathematical Center Tracts*.