A Near-Optimal Polynomial Time Algorithm for Learning in Certain Classes of Stochastic Games.

Ronen I. Brafman
Computer Science Department
Ben-Gurion University
Beer-Sheva, Israel

Moshe Tennenholtz
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, 32000 Israel

Abstract

We present a new algorithm for polynomial time learning of optimal behavior in single-controller stochastic games. This algorithm incorporates and integrates important recent results of Kearns and Singh [5] in reinforcement learning and of Monderer and Tennenholtz [7] in repeated games. In stochastic games, the agent must cope with the existence of an adversary whose actions can be arbitrary. In particular, this adversary can withhold information about the game matrix by refraining from (or rarely) performing certain actions. This forces upon us an exploration vs. exploitation dilemma more complex than in Markov decision processes in which, given information about particular parts of a game matrix, the agent must decide how much effort to invest in learning the unknown parts of the matrix. We present a polynomial time algorithm that addresses these issues in the context of the class of single controller stochastic games, providing the agent with near-optimal return.

1 Introduction

Stochastic games (SGs) extend Markov decision processes (MDPs) to a multi-agent environment. In classical stochastic games [9], two players, the *agent* and the *adversary*, engage in a series of competitive interactions. Thus, each state of an SG corresponds to a zero-sum game between the agent and the adversary, called a *stage-game* (or simply, a game, for short). Following each game, each of the players obtains some reward and both end up in a new state, corresponding to a new zero-sum game. The reward obtained by the players is a function of the current game (and hence, the current state) and their actions. Similarly, the new game is a stochastic function of the current game and the players' actions.

Much like in MDPs, the agent's goal is to find an optimal (or near-optimal) policy, i.e., a mapping from states (i.e., games) to actions. However, unlike in MDPs, such optimal policies are typically stochastic, i.e., each game is mapped to a probability distribution over actions rather than to a particular action. The definition of the optimization criteria for SGs is similar to that of MDPs, and include cumulative discounted reward, cumulative undiscounted reward (where the number of steps is finite but unbounded), average discounted reward, and average undiscounted reward; we concentrate on this last criterion. Unlike in MDPs, there need not be stationary optimal policies for infinite horizon stochastic games when the average undiscounted reward criterion is used, neither are there known polynomial time algorithms for computing solutions for such games. However, there is an important class of stochastic games in which stationary optimal policies in the infinite horizon average undiscounted reward case exist, and they can be computed in polynomial time. This type of game is called *single-controller stochastic game* (SCSG) [3, 8, 10], a name which derives from the fact that the state transitions depend on the action of the agent alone. Hence, the adversary's action influences the rewards only. In this paper, we concentrate on learning in SCSGs.

The learning algorithm that we present is based on Kearns and Singh's (KS) E^3 algorithm [5]. E^3 is a model-based learning algorithm (i.e., one in which a partial model of the MDP is formed) that has introduced a number of new concepts and ideas in the area of reinforcement learning. The extension of these ideas into SGs raises a number of issues that stem from the existence of an adversary whose behavior is unknown. In particular, this adversary can, at will, hide information from the agent by refraining from taking particular actions or by rarely playing such actions. This means that, unlike MDPs where the transition function and rewards can be made known to the agent with sufficient exploration, certain aspects of the model may never be known to the agent, or may take an unbounded time to learn. Therefore, we cannot emulate the two phase approach of the E^3 algorithm. There, the agent first attempts to learn enough about the model to obtain near-optimal return, after which it enters an exploitation phase. Instead, we will have to allow for the possibility of continuous learning.

Indeed, in stochastic games, a more complicated form of the exploration vs. exploitation problem arises. Recall that the exploration vs. exploitation dilemma refers to the question of whether to play optimally given the current knowledge, or to attempt to increase knowledge at the risk of unknown losses. KS solve this problem in the context of MDPs by using the fact that, if we know the value of the optimal policy of the MDP, we can, at each stage, examine whether we have learned enough to guarantee ourselves this value. Once this is the case, the agent needs no longer explore. Unfortunately, in SGs, this is not the case. Because of the ability of the adversary to hide parts of the game matrix, in many cases, we lack the information to calculate the value of a given policy. To overcome this, we employ techniques introduced by Monderer and Tennenholtz in the context of learning in repeated games [7]. Namely, we must explore at two levels. First, as in E^3, we must perform *global* exploration. That is, we must attempt to learn some facts about the different games and their associated transitions. In addition, we must perform *local* exploration as well, where we attempt to gain more information about our possibilities within a particular game.[1] Because it depends on

[1]Note that the notion of local/global exploration is sometimes used in the context single agent reinforcement learning literature with a different meaning. Here, we use the term *local* exploration to describe exploration within the game associated with a state, and *global* exploration to describe the exploration of

the behavior of the adversary, this local exploration part cannot be a-priori bounded. This is to be contrasted with the initial exploration phase of E^3, which takes polynomial time.

The algorithm presented in this paper addresses these and other issues, and yields near optimal performance for the agent in time polynomial in the basic problem parameters. It is, to the best of our knowledge, the first such result for any non-trivial class of stochastic games. Previous algorithms for learning in stochastic games [6, 4] were not concerned with analytic treatment and proof of efficiency, nor dealt explicitly with the exploration vs. exploitation issue in an efficient manner. However, Littman does provide asymptotic convergence results, and both papers deal with SGs in general. Indeed, the existence of polynomial-time near-optimal learning algorithms for SGs in general remains an important open question, and why our current approach fails when we move outside the scope of SCSGs.

In the following section we discuss single-controller stochastic games. In Section 3 we present our measure of complexity. In Section 4 we present our main theorem, which makes use of two basic ideas. A discussion of the first idea, based on a recent algorithm by Kearns and Singh [5] appears in Section 5. A discussion of the second idea, which is in the spirit of work on learning in repeated games, and in particular follows recent work by Monderer and Tennenholtz [7], is presented in Section 6. The synthesis of these ideas into a complete algorithm is presented in Section 7. In Section 8 we extended our results to general stochastic games, with some modifications. We conclude in Section 9. Appendix A contains a version of the Chernoff bound result [2, 1] upon which we rely heavily in various proofs.

2 Preliminaries

We first define *Single-Controller-Stochastic-Games* [SCSG]:

Definition 1 *A* single-controller-stochastic-game [SCSG] M *on states* $S = \{1, \dots, N\}$*, and actions* $A = \{a_1, \dots, a_k\}$*, consists of:*

- **Stage Games:** *each state* $s \in S$ *is associated with a zero-sum game in strategic form, where the action set of each agent is* A*. The first player is termed* agent *and the second player is termed* adversary.
- **Probabilistic Transition Function:** $P_M(s, t, a)$ *is the probability of a transition from* s *to* t *given that the first player (the* agent*) plays* a*.*

An SCSG is similar to an MDP. In both models, the agent's actions lead to transitions between states of the world. The main different is that the reward obtained by the agent for performing an action depends on its actions *and* the actions of the adversary. To model this, we associate a game with every state, and we will use the terms *state* and *game* interchangeably. Recall that a game in strategic form can be viewed as a matrix whose rows

the global state space.

correspond to the agent's actions and whose columns correspond to the adversary's actions. Thus, the agent controls the choice of the row, while the adversary controls the choice of the column. The entry in row i and column j in the game matrix contains the rewards obtained by the agent and the adversary if the agent plays his i^{th} action and the adversary plays his j^{th} action. In zero-sum games, the sum of these rewards (for any choice of i and j) is always 0. Here, we have made the simplifying assumption that the size of the action set of both the agent and the adversary is identical. The extension to sets of different sizes is trivial.

For ease of exposition we normalize the payoff of each stage game to be a non-negative real number between 0 and a constant P_{max} (i.e. the sum of the players' payoffs for any joint action is always P_{max}). We will also take the number of actions to be constant across different stage games.

For a given SCSG, the set of possible histories of length t is $(S \times A^2)^t$, and the set of possible histories, H, is the union of the sets of possible histories for all $t \geq 0$, where the set of possible histories of length 0 is S. [2] Given an SCSG, a policy for the agent is a mapping from H to the set of possible probability distributions over A. Hence, a policy determines the probability of choosing each particular action for each possible history. A *stationary policy* is a policy which depends only on S instead of on H. Such a policy associates with each state a probability distribution over the actions.

In order to define the notion of the *value* of a policy, we proceed as follows: Given an SCSG M and a natural number T, we denote the expected T-step undiscounted average reward of a policy π when the adversary follows a policy ρ, and where both π and ρ are executed starting from a state $s \in S$, by $U_M(s, \pi, \rho, T)$ (we omit subscripts denoting the SCSG when this causes no confusion). Notice that given policies for the agent and the adversary, an SCSG is transformed into a Markov chain (because the joint action for each game is fixed). This means that for a given SCSG M, every initial state s and policies π, ρ define a probability distribution over possible T-step histories. Denote this probability distribution by $pr_M^{s,\pi,\rho}$, and denote the average reward obtained by the agent in the course of some history h by $R(h)$. $U_M(s, \pi, \rho, T)$ can be formally defined as

$$U_M(s, \pi, \rho, T) = \Sigma_{T-step\ histories\ h} pr_M^{s,\pi,\rho}(h) R(h).$$

We continue by defining $U_M(s, \pi, T) = \min_{\rho\ is\ a\ policy} U_M(s, \pi, \rho, T)$, denoting the value that a policy π can guarantee in T steps starting from s. If $U_M(s, \pi, T)$ is independent of s we denote it by $U_M(\pi, T)$. We also denote $U_M(s, \pi) = \liminf_{T \to \infty} U_M(s, \pi, T)$, and $U_M(\pi) = \liminf_{T \to \infty} U_M(\pi, T)$ when the value is independent of s.

Finally, we will assume that the SCSG is *ergodic* in the sense that given any stationary policy of the agent, the probability of transition between each pair of states is greater than 0 regardless of the adversary behavior. This makes the value of each stationary policy well-defined (i.e., it is independent of the initial state). In particular, the *value* of M is the value $U_M(\pi^*)$ of an optimal policy π^* (and we know that for SCSGs, there is no loss of generality in assuming this policy is stationary).

[2] The notion of history is required in order to properly define the value of a policy. However, it does not play a part in the algorithm itself.

The ergodicity assumption is consistent with the treatment of [5], and is quite natural for the following reasons. Any Markov chain defined by a policy has one or more absorbing subsets of states. That is, subsets of the state space such that once the agent enters them, he will remain in them. In the initial stages of learning, the agent cannot be expected to know which policies will lead to which sets of absorbing states, and so we cannot really influence the choice of an absorbing state set. However, once we are within such a set, we would like to quickly learn how to behave. This is basically what we (and KS) offer.

3 Our Measure of Complexity

One of the important contributions of KS is the identification of the central parameter upon which the analysis of algorithms for learning in MDPs must be based, namely, the *mixing time*. Kearns and Singh [5] argue that it is unreasonable to refer to the efficiency of learning algorithms without referring to the efficiency of convergence to a desired value. They defined the *ϵ-return mixing time* of a stationary policy π to be the smallest value of T after which π *guarantees* an expected payoff of at least $U(\pi)-\epsilon$. More formally, in the context of SCSGs we say that a stationary policy π belongs to the set $\Pi(\epsilon, T)$ of stationary policies whose ϵ-return mixing time is at most T, if after time T, π returns an expected (average, undiscounted) payoff of at least $U(\pi) - \epsilon$ for *every* possible adversary behavior. That is, on the average, we have to employ T steps of policy π before our average accumulated reward is sufficiently close to the value of π. Notice that this means that an agent with perfect information about the nature of the games and the transition function will require at least T steps, on the average, in order to obtain a value (of almost) v using an optimal policy π whose ϵ-return mixing time is T. Clearly, one cannot expect an agent lacking this information to perform better.

Notice that the mixing time of a policy depends on the adversary's behavior: The agent's accumulated reward depends on his actions and on the adversary's actions. Hence, the adversary can influence the time it takes to obtain a particular sum of rewards, and we must take this into account. Luckily, in SCSGs, we can show that the mixing time of any policy is finite. This follows from the following observations. First, notice that given a stationary policy, if we disregard the payoffs obtained in each state, we get a standard Markov chain.[3] Now, given a particular stationary policy of the agent, the best response of the adversary is static, in the sense that it is built from best response actions at each state. If the adversary will use a different strategy from this best response strategy, the payoff of the agent will only increase. (Again, in SGs in general this is not true, since the adversary can influence the transition probabilities.) For a fixed policy of the adversary, it is possible to calculate the appropriate mixing time. As we have just observed, any deviation from this behavior will only increase the payoffs to the agent, thus decreasing the time needed to obtain the desired value.

We denote by $Opt(\Pi(\epsilon, T))$ the optimal expected undiscounted average return from among the policies in $\Pi(\epsilon, T)$. When looking for an optimal policy (with respect to policies

[3]This does not hold for SGs in general.

that mix at time T, for a given $\epsilon > 0$), we will be interested in approaching this value. In our algorithm, we shall assume that T and $Opt(\Pi(\epsilon, T))$ are given, and we shall attempt to learn a behavior whose average return is within ϵ from $Opt(\Pi(\epsilon, T))$. Once this problem is solved, we can, as in [5], remove these assumptions and measure the complexity as a function of the mixing time of the (stationary) optimal policy, where this value, as well as the value of the game are a-priori unknown.

4 The Main Theorem

Recall that we wish to obtain efficient behavior in SCSGs for agents that do not know the transition probabilities a-priori, nor do they know which games are associated with each state. We can show:

Theorem 1 *Given an SCSG M with N states and some $\epsilon > 0$, denote the stationary policies with ϵ-return mixing time T, by $\Pi_M(\epsilon, T)$, and the optimal expected return achievable by such policies by $Opt(\Pi_M(\epsilon, T))$. Let $0 < \delta, \phi < 1$ be constants. There exists an algorithm LSG, which takes $N, T, \epsilon, \delta, \phi$ as inputs, and is polynomial in $N, T, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{\phi}$, such that with probability $1 - \delta$ the expected return of LSG is greater than $(1 - \phi)(Opt_M(\Pi(\epsilon, T)) - 2\epsilon)$.*

Two features are immediately apparent from the statement of this theorem: We have to pay a small additive factor because of the fact that, as in MDPs, one cannot guarantee an optimal payoff within a bounded amount of time – a fact that holds even when we have an explicit, correct model. We also have to pay a multiplicative $1 - \phi$ factor which results from the ability of the adversary to hide desired information – in order to overcome this, we must perform some sampling throughout the process, which costs us a small, multiplicative loss.

The proof of this theorem relies on the following observations:

1. Since SCSGs share much structure with MDPs, we can adopt many aspects of Kearns and Singh's E^3 algorithm to this context. One of the fundamental ideas in this algorithm is the definition of a *known state,* i.e., a state for which the agent has accumulated sufficiently accurate statistics with regards to both rewards and transitions originating from it. We will adopt a similar notion, and utilize it similarly. However, in our case, the rewards for different actions depend on the actions taken by the adversary, who may act maliciously. Hence, careful adaptation of this notion is required for our context. Once this is done, we can follow the ideas of [5], regarding the exploitation vs. global exploration issue. This point is discussed in Section 5.

2. Suppose that we have been able to learn some information about a particular stage-game, and we attempt to employ an optimal policy with respect to this information. For example, suppose that we have learned two out of three columns of a given game, i.e., we have made sufficient observations of the outcome of all of our actions with two of the adversary's three actions. However, so far, we do not have knowledge of

what happens when the adversary chooses this third action. Furthermore, suppose that some action a is the optimal action when the adversary sticks to the two actions we are familiar with. We have the following dilemma: we could try to exploit our current information about the game, and decide to play a. However, if the adversary chooses to start playing his third action now, and a is inadequate against this action, we will attain low returns. On the other hand, we could decide that we need to explore the whole game first, playing randomly until we learn enough about the third column. However, we have no guarantee that the adversary will actually play the third action and so we may learn nothing new while continuously attaining sub-optimal returns. This issue is what we referred to earlier as the *local* explore vs. exploit problem, and our main problem will be to find some reasonable tradeoff between these two options, one that will allow us to achieve the desired level of performance. This issue is discussed in Section 6.

5 Exploitation with Global Exploration

We will make use of the following definition, which enables us to state a version of KS's simulation Lemma [5] in the context of stochastic games.

Definition 2 *Let M and $\bar{M}$ be SCSGs over the same state space. We say that $\bar{M}$ is an α-approximation of M if for every state s we have:*

1. *If $P_M(s,t,a)$ and $P_{\bar{M}}(s,t,a)$ are the probabilities of transition to t given that the action carried out by the agent is a, in M and $\bar{M}$ respectively, then, $P_M(s,t,a) - \alpha \leq P_{\bar{M}}(s,t,a) \leq P_M(s,t,a) + \alpha$*
2. *The game associated with s in $\bar{M}$ is the game associated with it in M restricted to a non-empty subset of the columns.*

Lemma 1 *Let M and $\bar{M}$ be any SCSGs over N states, where $\bar{M}$ is an $O(\frac{\epsilon}{NTP_{max}})^2$-approximation of M, then for every state s and policy π we have that $U_{\bar{M}}(s,\pi,T) \geq U_M(s,\pi,T) - \epsilon$.*

Proof: First, we prove the Lemma with the following modification: instead of claiming the above for every state s and policy π, we will claim it for every state s, policy π, and fixed adversary policy ρ. Clearly, if this holds, then the actual statement holds given that $U_M(s,\pi,T) \stackrel{\text{def}}{=} \min_\rho U_M(s,\pi,e,T)$. When the adversary's policy is fixed, an SCSG reduces to an MDP. If we make the additional assumption that M and $\bar{M}$ have the same number of columns in every game, then our claim reduces to Kearns and Singh's Simulation Lemma [5]. However, we allow for $\bar{M}$ to contain games that correspond to games from M, but where the set of columns corresponds to a strict subset of the set of adversary's actions. However, any restriction of the adversary's actions can only result in higher payoffs for the agent. ∎

The last argument in the proof of Lemma 1 may lead the reader to think that the second item in definition 2 plays an insignificant role. However, as we now discuss, this aspect of the definition relates to the major concept of this section, that of *known states.*

We say that a state is *known* if we know its associated transition probabilities within a range of $O(\frac{\epsilon}{NTP_{max}})^2$ from their true value, and we know *at least one column* in it. This means that the agent knows the transition probabilities associated with each of its actions, and, in addition, it knows the values appearing in at least one column of the game matrix. Notice that a state may be considered known even though we do *not* have complete empirical evidence about the game associated with that state. This point is crucial, and differs from the situation in Kearns and Singh's proof.

In what follows we will define the notion of an induced SCSG. Intuitively, the induced SCSG corresponds to the agent's current model of the true SCSG. This model is based on the set of known states and contains an additional special state representing the set of all unknown states. Then, in Lemma 2, we will show that for an induced SCSG, there is always either a policy that attains near-optimal payoff or a policy that has some fixed minimal probability of reaching the unknown state, and hence, of leading to new information.

Definition 3 *Let M be an SCSG, and let L be any subset of the set of states S. The* induced SCSG, *M_L, has states $L \cup \{l_0\}$, and transitions and state games as follows:*

1. *The states in M_L are associated with the same games as in M.*
2. *The state l_0 is associated with a game where the adversary obtains the value P_{max} for any joint action (a "worst case" state for the agent).*
3. *For any action a, $P_{M_L}(l_0, l_0, a) = 1$.*
4. *For any states $s, t \in L$, and $a \in A$, we have that $P_M(s, t, a) = P_{M_L}(s, t, a)$.*
5. *For every $s \in L$, and $t \notin L$, and for every action $a \in A$, we have that $P_{M_L}(s, t, a) = 0$.*
6. *For every $s \in L$, and $a \in A$, we have that $P_{M_L}(s, l_0, a) = \Sigma_{j \notin L} P_M(s, t, a)$.*

What follows is the SCSG version of [5]'s "Explore or Exploit" lemma.

Lemma 2 *Let M be an SCSG. Let L be any subset of S, and let M_L be the corresponding induced SCSG. Then, for any $s \in L$, and for every T and $0 < \alpha < 1$, we have that there exists a policy π in M_L such that $U_{M_L}(s, \pi, T) \geq U_M(s, \pi^*, T) - \alpha$ (where π^* denotes the corresponding T-step optimal policy in M), or there exists a policy in M_L such that the probability that a walk of T steps of that policy will reach l_0 exceeds $\frac{\alpha}{P_{max}}$ for every adversary behavior.*

Aside from minor modifications, the proof of [5] carries over to this case as well. However, because of its importance, and because it explains how to perform exploration, we shall repeat it here.

Let π be a policy for M satisfying $U_M(s,\pi,T) = U_M(s,\pi^*,T)$, and suppose that $U_{M_L}(s,\pi,T) < U_M(s,\pi^*,T) - \alpha$ (otherwise, π is already a witness to the claim of the lemma). We may write:

$$U_M(s,\pi,T) = \sum_p Pr^{\pi}_M[p]U_M(p) = \sum_q Pr^{\pi}_M[q]U_M(q) + \sum_r Pr^{\pi}_M[r]U_M(r)$$

where the sums are over, respectively, all T-paths p in M, all T-paths q in M in which every state in q is in L, and all T-path r in M in which at least one state is not in L. Keeping this interpretation of the variables p, q and r fixed we may write $\sum_q Pr^{\pi}_M[q]U_M(q) \leq \sum_q Pr^{\pi}_{M_L}[q]U_{M_L}(q) \leq U_{M_L}(s,\pi,T)$. The first inequality follows from the fact that for any path q in which every state is in L, $Pr^{\pi}_M(q) = Pr^{\pi}_{M_L}(q)$ and $U_M(q) \leq U_{M_S}(q)$. (This latter inequality is the only place where this proof differs from [5].) This latter inequality stems from the fact that in M_L, the adversary may have fewer options to choose from (because some columns were deleted) and therefore, his payoff may decrease. The last inequality stems from the fact that $U_{M_L}(s,\pi,T)$ takes the sum over all T-paths in M_L, not just those that avoid the absorbing state l_0. Thus we obtain that $\sum_q Pr^{\pi}_M[q]U_M(q) \leq U_M(s,\pi^*,T) - \alpha$, which implies that $\sum_r Pr^{\pi}_M[r]U_M(r) \geq \alpha$. But $\sum_r Pr^{\pi}_M[r]U_M(r) \leq P_{max}\sum_r Pr^{\pi}_M(r)$, and so $\sum_r Pr^{\pi}_M[r] \geq \alpha/P_{max}$. ∎

It is now natural to look at $\bar{M}_L$, where L is the set of known states. The Lemma presented above tell us that if at each known state the adversary performs only actions that are associated with fully known columns, then one can either find a policy which, if run for T steps, attains the desired value (i.e., an exploitation policy), or a policy that, with sufficient probability, will reach some unknown state (i.e., an exploration policy). In fact, Lemma 2 shows that the optimal T-step policy for M is always either an exploration policy or an exploitation policy for $\bar{M}_L$.

In order to utilize this result we can proceed as follows: Using Lemma 1 and the fact that we know the SG's value, we can compute the optimal policy for $\bar{M}_L$ and determine whether it is near-optimal. If not, we choose to explore. We can come up with a good exploration policy by computing an optimal policy for $\hat{M}_L$, where $\hat{M}_L$ is defined much as $\bar{M}_L$, but such that in l_0 the adversary obtains 0 reward for every joint action, and in L the adversary obtains P_{max} for every joint action. This policy is the policy with the highest probability of attaining a non-L state within T steps. The Explore and Exploit Lemma guarantees that if our initial attempt to find an exploitation policy fails, then this policy will explore with sufficient probability.

We note in passing that this proof of the Explore or Exploit Lemma cannot be extended effectively to handle SGs. It does imply that the optimal policy for M is either an exploration or an exploitation policy for every adversary behavior. But we do not know the optimal policy for M. Assuming we know the value of the SG, we can, as indicated above, check whether its counterpart, the optimal policy for $\bar{M}_L$, is good enough. However, it is possible for this latter policy to be good enough given certain adversary behaviors and insufficient given other adversary behaviors. If that is the case, then we do not have a guaranteed exploration policy nor do we have a guaranteed exploitation policy.

Let us sum up the observations and ideas discussed so far. The basic idea is to define a known state as a state which has been visited enough times to collect meaningful statistical data on at least one column of the game matrix. When an agent reaches an unknown state it will *randomly* select an action and execute it. (Notice that we cannot employ the balanced wandering approach of KS, since an adversary can exploit this to prevent learning). After polynomially many steps, at least one state will become known with sufficient probability.

When an agent reaches a known state, it will perform exploration or exploitation depending on the considerations described above. This requires the computation of the optimal expected T-steps return in $\bar{M}_L$, where L is the set of known states, and this computation must be repeated each time a new state becomes known or a new column becomes known. This computation can be performed in polynomial time (as a function of N and T) using standard dynamic programming updates. If the attained value is not close enough to the desired value we will generate an exploration step in order to reach an unknown state and make additional random sampling of actions.

Notice that there are only polynomially many parameters to learn (one for each joint action in each state). After a number of successful explorations that is polynomial in the problem parameters, we are guaranteed to have an exploitation policy. Of course, not every exploration attempt is successful, rather each has a probability of at least α/P_{max} of succeeding. Using standard Chernoff bounds (see Appendix A) with high probability, after a polynomial (in α/P_{max}) number of explorations, all entries will be known. By choosing a value for α that is polynomial in the problem parameters, we deduce that after polynomially many stages, with high probability, all states become known or the value is obtained. (See Section 7 for a more formal treatment of these observations).

Unfortunately, the above analysis (which works for MDPs) is inadequate for SCSGs. The reason is that in defining $\bar{M}_L$ we considered only known columns of known games. That is, we have implicitly assumed that in each game the adversary will execute actions that correspond to known columns. Naturally, the adversary is free to play actions corresponding to unknown columns. This can lead to lower payoffs. In the following section we explain how this problem is dealt with.

6 Exploitation with Local Exploration

In the previous sections we modified the definition of a *known* state introduced by KS in order to adapt their techniques to the context of stochastic games. However, we saw that an adversary can exploit the existence of unknown columns within a known state, if we are not careful. To handle this problem, we adopt techniques from the work of Monderer and Tennenholtz [7] on learning in repeated games.

Lemma 3 *Consider an SCSG M with N states and k actions, where the transition probabilities are known and at least one column at each state is (fully) known. Consider the* *policies whose ϵ-return mixing time is T, and let $0 < \theta, \gamma < 1$. There exists an algorithm polynomial in $T, N, \frac{1}{\theta}, \frac{1}{\gamma}, \frac{1}{\epsilon}$ such that its execution for polynomially many steps leads to an expected average payoff of $(1-\theta)(Opt(\Pi(\epsilon, T)) - \epsilon)$ with probability of at least $1-\gamma$.*

Proof: The algorithm runs in stages that last at most T-steps. At each such stage, a T-step optimal policy with respect to known columns only, π, is computed. We know that π has an expected payoff of at least $Opt(\Pi(\epsilon, T) - \epsilon)$, *provided* that the adversary selects only actions which correspond to known columns. However, as there is no guarantee that the adversary will act this way, we execute a policy π_m instead of π. π_m is equivalent to π with probability of $1 - \frac{\theta}{2}$, and with probability of $\frac{\theta}{2}$ it randomly selects a time i between 1 and T and an action, and when the i-th step in the execution is reached it executes this randomly chosen action.

A stage is considered finished if either: (1) the policy has been executed for T steps, or (2) either the adversary used an action that corresponds to an unknown column or the agent deviated from π.

It can be shown that the probability that unknown columns will be selected $(\frac{2TkNX}{\theta})^3 k^2 N$ times before all entries are known is less than $k^2 N e^{\frac{-2kNTX}{\theta}}$. This follows from the fact that (1) the probability of failure of learning all columns in at least one set (from among k^2N sets) of $(\frac{2TkNX}{\theta})^3$ adversary deviations (i.e. selections of unknown columns) is smaller than k^2N times the probability of failing to learn in one such set of deviations, and that (2) there are at most k^2N entries to learn.

More specifically, the probability that an unknown entry will be learned by the agent when the adversary selects a column which corresponds to at unknown entry is at least: $\frac{\theta}{2}\frac{1}{T}\frac{1}{k}$. This probability is greater than $\frac{\theta}{2TkNX}$ where $X > 1$ (the actual value of X will be determined as explained below). This implies that the probability that no new entry will be learned after $(\frac{2kTNX}{\theta})^3$ selections of unknown columns by the adversary is lower than $(1 - \frac{1}{\frac{2kTNX}{\theta}})^{(\frac{2kTNX}{\theta})^3}$, which is by itself lower than $e^{\frac{-2kTNX}{\theta}}$. Consider $(\frac{2TkNX}{\theta})^3 k^2 N$ selections by the adversary of columns with unknown entries, partitioned into k^2N sets of equal size, each of which includes $(\frac{2TkNX}{\theta})^3$ such selections. The chances that at least one entry will remain unknown, among the k^2N entries, is bounded by k^2N times the probability of not learning the value of an additional entry along $(\frac{2TkNX}{\theta})^3$ such selections. This implies that after $(\frac{2TkNX}{\theta})^3 k^2 N$ times columns with unknown entries have been selected, all entries in all games will be exposed, with probability which is at least $1 - k^2 N e^{\frac{-2kTNX}{\theta}}$.

We run the the algorithm for Y stages, such that $\frac{(\frac{2TkNX}{\theta})^3 k^2 N}{Y} < \frac{\theta}{2}$ and Y is polynomial in the problem parameters. This will guarantee that the proportion of stages in which we do not follow π is smaller than $\theta/2$. Hence, we must have $Y > (\frac{2TkNX}{\theta})^3 k^2 N \frac{2}{\theta}$. We also need to require that $k^2 N e^{\frac{-2kNTX}{\theta}} < \gamma$. Hence, $e^{\frac{2kNTX}{\theta}} > \frac{k^2N}{\gamma}$, or $X > \log(\frac{k^2N}{\gamma})\frac{\theta}{2kNT}$. As can be directly observed from these inequalities, we can indeed choose (polynomial) X and Y that satisfy these conditions.

To complete the proof we need to show that we obtain the desired expected value. This follows from the fact that after Y stages (with the corresponding probability) only at most

$\frac{\theta}{2}$ of the stages correspond to adversary deviations, while in $1 - \frac{\theta}{2}$ of the stages an expected payoff of $(1 - \frac{\theta}{2})(Opt(\Pi(\epsilon, T)) - \epsilon)$ is obtained. ∎

Hence, once we have reached a situation where we have a policy that can obtain the desired value if the adversary behaves "nicely", we can modify this policy to a policy which obtains almost the desired value or learns a new fact about the states (with overwhelming probability). Thus, we tradeoff some exploitation for exploration in a manner that guarantees that if the adversary plays an unknown column polynomially many times, we will learn this column after a polynomial number of steps. If the adversary rarely plays that column, we will rarely encounter it, and so the possible losses stemming from the randomization effect are almost surely insignificant given a sufficiently long (but polynomial) number of steps.

7 The Algorithm and Its Analysis

The LSG algorithm can be executed for any desired number of steps $t < \infty$. For sufficiently large values of t (polynomial in the problem parameters) a near optimal average return is guaranteed, as stated in Theorem 1. The algorithm consists of the following steps:

1. Initialize the set L of known states to be empty.
2. If the current state is not in L:
   (a) Randomly sample an action and execute it.
   (b) If following this sampling the current state has been visited enough times (see Definition 2, Lemma 1, and the discussion after Lemma 1), and *at least* one column of the game associated with it is known, add it to the set of known states L.
3. If the current state is in L (i.e., a known state), perform an off-line computation on $\bar{M}_L$ in order to check whether a value of at least $Opt(\Pi_M(T, \epsilon)) - \epsilon$ can be guaranteed, *assuming the adversary uses only actions that correspond to fully known columns.*
4. If such a value can be guaranteed by a policy π, then the policy π_m is executed.[4] The run of π_m is halted whenever a deviation of the adversary from the actions associated with fully known columns is observed, when the agent deviates from π, when we have reached an unknown state, or when a new column in a state in L becomes fully known.
5. Otherwise, a payoff of $Opt(\Pi_M(T, \epsilon)) - \epsilon$ can not be obtained, and a (global) exploration policy π' is executed (see Section 5) for T steps or until an unknown state is reached. This policy is guaranteed by Lemma 2 to reach a state outside L with probability of at least $\frac{\epsilon}{P_{max}}$ within T steps.

[4]Recall that π_m performs the optimal policy with respect to known states with some amount of local exploration.

6. In all cases, whenever an entry in a stage game is learned, the value of it is kept in memory.

We now prove Theorem 1 by showing that LSG has the desired properties. In the course of the proof we also establish more precise bounds on its running time.

Theorem 2 *The LSG algorithm attains an average reward of* $(1-\phi)(Opt_M(\Pi(\epsilon,T))-2\epsilon)$ *in time polynomial in* $N, T, \epsilon, \delta, \phi$ *with probability of at least* $1-\delta$.

Proof: First, notice that the algorithm's execution can be separated into two stages. In the first stage, we do not have an exploitation policy, while in the second stage we have an exploitation policy. We claim that (with high probability) each such stage will take a polynomial number of steps.

Consider the first stage in which we do not have an exploitation policy. This stage will end at least as soon as we know all the model parameters. As explained following Lemma 2, there is a polynomial number of parameters to learn. Each execution of the exploration policy leads to learning one such parameter within T steps with a probability polynomial in the problem parameters (see below). Using standard Chernoff bound analysis, it immediately follows that with sufficiently high probability we will execute at most a polynomial number of T-step explorations before all parameters are known. As for the second, exploitation stage, Lemma 3 guarantees that (with sufficient probability) it will take a polynomial number of steps.

Next, consider the expected payoff. We have provided a polynomial bound on the number of exploration steps. We have shown, in Lemma 3 that the exploitation stage will attain near-optimal reward. We can now simply repeat the exploitation stage for a sufficiently large (polynomial) number of steps so that the (bounded) losses of the initial exploration stage are compensated for by the (sufficiently large) exploitation stage.

To provide a more explicit bound on the running times and the failure probability, we now consider the five sources of failure of our algorithm. We will take steps to ensure that the probability of failure for any one of these reasons is no more than $\delta/5$, yielding the desired conclusion.

1. In some states the algorithm may have a poor estimate of the true next-state distribution. As we established earlier, we need to know the transition probabilities within $(\frac{\epsilon}{NTP_{max}})^2$. Consider a set of trials, where we select action a in state s. Consider the probability of moving from state s to state t given action a in a given trial, and denote it by p. Notice that there are Nk^2 such probabilities (one for each game and pair of agent-adversary actions). Therefore, we would like to show that the probability of failure in estimating p is less than $\frac{\delta}{5Nk^2}$. Let X_i be an indicator random variable, that is 1 iff we moved to state t when we were in state s and selected an action a in trial i. Let $Z_i = X_i - p$. Then $E(Z_i) = 0$, and $|Z_i| \leq 1$. Then, Chernoff bound implies that (for any K_1) $Prob(\Sigma_{i=1}^{K_1} Z_i > {K_1}^{\frac{2}{3}}) < e^{\frac{-{K_1}^{\frac{1}{3}}}{2}}$. This implies that $Prob(\frac{\Sigma_{i=1}^{K_1} X_i}{K_1} - p >$

${K_1}^{-\frac{1}{3}}) < e^{\frac{-{K_1}^{\frac{1}{3}}}{2}}$. Similarly, we can define $Z'_i = p - X_i$, and get by Chernoff bound that $Prob(\Sigma_{i=1}^{K_1} Z'_i > {K_1}^{\frac{2}{3}}) < e^{\frac{-{K_1}^{\frac{1}{3}}}{2}}$. This implies that $Prob(p - \frac{\Sigma_{i=1}^{K_1} X_i}{K_1} > {K_1}^{-\frac{1}{3}}) < e^{\frac{-{K_1}^{\frac{1}{3}}}{2}}$. Hence, we get that $Prob(|\frac{\Sigma_{i=1}^{K_1} X_i}{K_1} - p| > {K_1}^{-\frac{1}{3}}) < 2e^{\frac{-{K_1}^{\frac{1}{3}}}{2}}$.

We now choose K_1 such that ${K_1}^{-\frac{1}{3}} < (\frac{\epsilon}{NTP_{max}})^2$, and $2e^{\frac{-{K_1}^{\frac{1}{3}}}{2}} < \frac{\delta}{5Nk^2}$. This is obtained by taking $K_1 = max((\frac{NTP_{max}}{\epsilon})^6, -8ln^3(\frac{\delta}{10Nk^2})) + 1$.

The above guarantees that we fail in getting the desired approximation with a probability which is less than $\frac{\delta}{5}$.

2. When sampling a random action in state s, we have only a probability of $\frac{1}{k}$ of hitting the action a. We need to get K_1 hits of action a in order to get the desired statistics. Therefore, we show that after some K_2 attempts per stage game, where K_2 is polynomial in K_1 we will obtain the desired K_1 hits with probability which is no less than $1 - \frac{\delta}{5kN}$. Since there are N states and k actions, this will give us a failure probability of less than $\frac{\delta}{5}$ due to this reason. Let X_i be an indicator random variable, where $X_i = 1$ iff we select a in the state s, and 0 if we select another action in that state. Let $Z_i = X_i - \frac{1}{k}$, and $Z'_i = \frac{1}{k} - X_i$, and apply Chernoff bound on the sum of Z_i's and Z'_is as before. We get: $Prob(|\Sigma_{i=1}^{K_2} X_i - K_2 k| > K_2^{\frac{1}{3}}) < 2e^{\frac{-{K_2}^{\frac{1}{3}}}{2}}$. The desired result is now obtained by choosing K_2 such that $K_2 k + K_2^{\frac{1}{3}} > K_1$, and $2e^{\frac{-{K_2}^{\frac{1}{3}}}{2}} < \frac{\delta}{5kN}$.

3. The previous analysis implies that after no more than NK_2 visits to the set of unknown states, we will be able to learn an unknown state with an appropriately small probability of failure. Actually, the above result gives a per-state bound, but using the pigeon-hole principle, we extend it to a set of states and denote the required bound by $K_3 (= K_2 N)$. The Exploit or Explore Lemma gives a probability of $\frac{\alpha}{P_{max}}$ of getting to explore. We now wish to show that after K_4 attempts to explore (i.e. when we do not exploit), we obtain the K_3 required visits. Let X_i be an indicator random variable which is 1 if we reach to the exploration state (l_0 in Lemma 2) when we do not exploit, and 0 otherwise. Let $Z_i = X_i - \frac{\alpha}{P_{max}}$, and let $Z'_i = \frac{\alpha}{P_{max}} - X_i$, and apply Chernoff bound on the sum of Z_i's and Z'_is as before. We get that $Prob(|\Sigma_{i=1}^{K_4} X_i - \frac{K_4 \alpha}{P_{max}}| > K_4^{\frac{1}{3}}) < 2e^{\frac{-{K_4}^{\frac{1}{3}}}{2}}$. We can now choose K_4 such that $K_4^{\frac{1}{3}} + K\frac{\alpha}{P_{max}} > K'_3$ and $2e^{\frac{-{K_4}^{\frac{1}{3}}}{2}} < \frac{\delta}{5k^2N}$ to guarantee that we will have a failure probability of less than $\frac{\delta}{5}$ due to this reason.

4. When we perform T-step exploitation with no local exploration we reach an expected return of $Opt(\Pi_M(T,\epsilon)) - \epsilon$, but the actual return may be lower. This point is handled by the fact that after polynomially many local exploitations are carried out, $Opt(\Pi_M(T,\epsilon)) - \frac{3}{2}\epsilon$ can be obtained with a probability of failure of at most $\frac{\delta}{5}$. This is obtained by standard Chernoff bounds, and makes use of the fact that the standard deviation of the expected reward in a T-step policy is bounded because the maximal reward is bounded by P_{max}. More specifically, consider $z = MnT$ exploitation stages

for some $M > 0$. Denote the average return in an exploitation stage by μ, and let X_i denote the return in the i-th exploitation stage ($1 \leq i \leq z$). Let $Y_i = \frac{\mu - X_i}{P_{max}}$. Notice that $|Y_i| \leq 1$, and that $E(Y_i) = 0$. Chernoff bound implies that: $Prob(\Sigma_{j=1}^{z} Y_j > z^{\frac{2}{3}}) < e^{\frac{-z^{\frac{1}{3}}}{2}}$ This implies that the average return along z iterations is at most $\frac{P_{max}}{z^{\frac{1}{3}}}$ lower than μ with probability of at least $e^{\frac{-z^{\frac{1}{3}}}{2}}$. By choosing M such that $z > (\frac{2P_{max}}{\epsilon})^3$, and $z > 8(ln(\frac{\delta}{5}))^{-3}$, we get the desired result: with probability less than $\frac{\delta}{5}$ the value obtained will not be more than $\frac{\epsilon}{2}$ lower than the expected value.

5. The agent may get a low payoff because it does not know the entries in some column and does not learn new entries in unknown columns. This is handled by Lemma 3, where we can choose the failure probability, γ, as needed.

By making the failure probability less than $\frac{\delta}{5}$ for each of the above stages, we are able to get the desired result.

From the proof, we can also observe the bounds on running times required to obtain this result. First, the exploration stage requires no more than K_4 steps, and we have seen that K_4 is polynomial in the problem parameters. Notice that in practice, the only bound that the implementer of the algorithm needs to maintain is the K_1 bound on the sample size used to estimate the transition probabilities. Next, we can commence the exploitation stage. Here, we need to ensure that the number of steps performed is greater than both the parameter z described in item (4) above and the parameter Y described in Lemma 3. In addition, we must compensate for the potential losses from the exploration stage. That is, if we explored for K steps (where we know that $K \leq K_4$), then the exploitation stage must last at least K' steps, where $P_{max}K/K' \leq \epsilon$.

Finally, we remove the assumptions that both the value and its ϵ-return mixing time are known. This is straightforward and almost identical to the treatment given by [5]. First, as to knowledge of the value, this is needed when we have to decide whether to explore or exploit. Lemma 2 states that we can either get enough return or we have a sufficiently high probability of reaching a new state quickly. We can calculate the probability of reaching a new state quickly without knowledge of the value. Whenever that probability is greater than the desired bound, we shall explore. As explained earlier, with overwhelming probability, we shall remain with known states only after polynomially many exploration phases. If the probability of reaching a new state quickly is small, we can immediately deduce that the optimal policy for our current, approximate model is indeed a near-optimal policy.

Next, we must deal with the lack of knowledge of T. The idea is as follows: from the proof of the algorithm's properties, one can deduce some polynomial P in the problem parameters such that if T is the mixing-time, then after $P(T)$ steps we are guaranteed, with probability $1 - \delta$, the desirable return. Hence, we can simply attempt to run this algorithm for $T = 1, 2, 3, \ldots$. For each value of T, we run the algorithm $P(T)$ time. Suppose that T_0 is the mixing time, then after $O(P(T_0)^2)$ steps, we will obtain the desirable return. ∎

One thing to notice is that this algorithm does not have a final halting time and will be applied continuously as long as the agent is functioning in its environment. The only caveat is that at some point our current mixing time candidate T will be exponential in the actual mixing time T_0, at which point each step of the algorithm will require an exponential calculation. However, this will occur only after an exponential number of steps. This is true for the E^3 algorithm too.

Another point worth mentioning is that in SCSGs, the agent may never know some of the columns. Consequently, if π is the optimal policy given full information about the game, the agent may actually converge to a policy π' that differs from π, but which yields the best return given the adversary's actual behavior. This return will be no smaller than the return guaranteed by π. The mixing time of π' will, in general, differ from the mixing time of π. However, we are guaranteed that if T_0 is the ϵ-return mixing time of π, and v is its value, after time polynomial in T_0, the agent's actual return will be at least v (subject to the deviations afforded by the theorem).

8 Conclusion

We described an algorithm for learning in single-controller stochastic games. This algorithm extends earlier work of Kearns and Singh [5] on learning in MDPs using the techniques of Monderer and Tennenholtz for learning in repeated games [7]. In describing the algorithm we aimed for clarity rather than efficiency, with the sole constraint of providing a polynomial time algorithm. A more careful analysis will lead to reduced running time.

Extension of this result to SGs in general remains an open problem. Unfortunately, a straightforward extension of the results presented here is not possible. In SGs, stationary equilibria do not always exist, nor are there known polynomial time solution algorithms for SGs in general. In fact, it is not known whether SGs have finite mixing time. This first fact implies that our approach for defining the value of an SCSG does not work for SGs in general because we cannot restrict our attention to stationary policies, and hence, we must take into account the initial state. Of course, we can choose to explicitly restrict out attention to stationary policies for the agent and attempt to learn a policy that is optimal with respect to this class of policies. This task seems feasible, but cannot be accomplished by a straightforward extension of the Explore or Exploit Lemma for reasons described in Section 5.

Finally, we remark that when we wish to model an agent acting in a non-stochastic environment, an alternative model to an MDP would be an SG in which nature is taken as the adversary. Hence, in appropriate circumstances our algorithm can be used to learn to behave in non-stochastic environments.

A Chernoff Bounds

In our analysis we make use of the following powerful result:

Theorem 3 (Chernoff) *Let* X_i, $1 \leq i \leq n$ *be mutually independent with all* $E(X_i)$*=0 and all* $|X_i| \leq 1$. *Let* $S = X_1 + X_2 + \cdots + X_n$. *Then,* $Pr[S > a] < e^{\frac{-a^2}{2n}}$.

By appropriate normalization the above result can be applied in a straightforward manner to X_is with some constant expectation and bounds on their values. This is indeed the case in our study; this is easily implied by the fact the entries in the game matrices are taken to be bounded.

To illustrate the way that the bound is used, consider a sequence of n trials (e.g. exploitations which use an optimal policy), where X_i measures the deviation from desired quantity, C, at trial i (e.g. the deviation from the expected optimal payoff). Now take for example $a = n^{\frac{2}{3}}$, then we get that $Pr[S > n^{\frac{2}{3}}] < e^{-n^{\frac{1}{3}}}$. Hence, with overwhelming probability (exponential in n), we get that the deviation of the actual average of the X_is from the desired value is negligible (and smaller or equal to $n^{-\frac{1}{3}} = \frac{n^{\frac{2}{3}}}{n}$.).

This implies that if we are able to obtain a desired expected value, then by polynomially many repetitions we will be able to actually obtain it with overwhelming probability.

Acknowledgments: We are grateful to the anonymous reviewers for their important comments. The first author was supported in part by the Paul Ivanier Center for Robotics Research and Production Management. The second author was supported by the US-Israel Binational Science Foundation.

References

[1] N. Alon, J.H. Spencer, and P. Erdos. *The Probabilistic Method.* John Wiley & Sons, 1992.

[2] H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[3] A. Horijk and L.C.M. Kallenberg. Linear Programming and Markov Games. In O. Moeschlin, editor, *Game Theory and Mathematical Economics*, pages 307–319. North Holland, 1981.

[4] J. Hu and M.P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, 1998.

[5] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *International Conference on Machine Learning*, 1998.

[6] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th Intl. Conf. on Machine Learning*, pages 157–163, 1994.

[7] D. Monderer and M. Tennenholtz. Dynamic Non-Bayesian Decision-Making. *Journal of Artificial Intelligence Research*, 7:231–248, 1997.

[8] T. Parthasarathy and T.E.S. Raghavan. An Order Field Property for Stochastic Games when One Player Controls Transition Probabilities. *J. Optim. Thoery Appl.*, 33:375–392, 1981.

[9] L.S. Shapley. Stochastic Games. In *Proc. Nat. Acad. Scie. USA*, volume 39, pages 1095–1100, 1953.

[10] O.J. Vrieze. Linear Programming and Undiscounted Stochastic Game in which One Player Controls Transitions. *OR Spektrum*, 3:29–35, 1981.