A Near-Optimal Poly-Time Algorithm for Learning in a Class of Stochastic Games

Ronen I. Brafman
Dept. of Math and Computer Science
Ben-Gurion University
Beer-Sheva, Israel
braftnan@cs.bgu.ac.il

Moshe Tennenholtz
Faculty of Industrial Eng. and Management
Technion
Haifa, 32000 Israel
moshet@ie.technion.ac.il

Abstract

We present a new algorithm for polynomial time learning of near optimal behavior in stochastic games. This algorithm incorporates and integrates important recent results of Kearns and Singh [1998] in reinforcement learning and of Monderer and Tennenholtz [1997] in repeated games. In stochastic games we face an exploration vs. exploitation dilemma more complex than in Markov decision processes. Namely, given information about particular parts of a game matrix, how much effort should the agent invest in learning its unknown parts. We explain and address these issues within the class of single controller stochastic games. This solution can be extended to stochastic games in general.

1 Introduction

Stochastic games (SGs) extend Markov decision processes (MDPs) to a multi-agent environment. In classical stochastic games [Shapley, 1953], two players, the *agent* and the *adversary,* engage in a series of competitive interactions. Thus, each state in an SG is associated with a game between the agent and the adversary. Following each game, each of the players obtains some reward and both end up in a new state (i.e., game). The reward obtained by the players is a function of the current state and their actions; the new state 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 mixed, i.e., each game is mapped to a probability distribution over actions rather than to a particular action. The optimization criteria for SGs are similar to those used in 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. Unfortunately, it is not known whether *stationary* optimal policies for infinite horizon stochastic games exist when the average undiscounted reward criterion is used. Moreover, there are no 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 games is called *single-controller stochastic games* (SCSGs) [Horijk and Kallenberg, 1981; Parthasarathy and Raghavan, 1981; Vrieze, 1981], a name which derives from the fact that the state (or game) 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 containing zero-sum games. The extension of these results into zero-sum stochastic games is discussed in the full version of this paper [Brafman and Tennenholtz, 1998].

The learning algorithm we present is based on the E^3 algorithm [Kearns and Singh, 1998], 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. Therefore, certain aspects of the model may never be known to the agent or may take an unbounded time to learn, unlike in MDPs. Therefore, we cannot emulate the two phase approach of the E^3 algorithm. There, die 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. Kearns and Singh (KS) [Kearns and Singh, 1998] 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 die 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 Mon* derer and Tennenholtz [1997] in the context of learning in repeated games. Namely, we explore at two levels. First, as in E^3, we perform *global* exploration. That is, we attempt to learn some facts about different games. In addition, we perform *local* exploration in order to extend our knowledge about particular games. Because it depends on 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 2?[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 in the context of stochastic games. Previous algorithms for learning in SGs [Littman, 1994; Hu and Wellman, 1998] 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.

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 [1998] 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 [1997], is presented in Section 6. The synthesis of these ideas into a complete algorithm is presented in Section 7. We conclude in Section 8. The full version of this papa:, [Brafman and Tennenholtz, 1998], contains all proofs, explains the issues that general stochastic games pose, and explains how the results presented here can be extended to the general case.

2 Preliminaries

First, we define *Single-Controller-Stochastic-Games* (SCSG):

Definition 1 *A* single-controller-stochastic-game *M on states* $S = \{1, \ldots, N\}$, *and actions* $A = \{a_1, \ldots, a_k\}$, *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 player 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 tmnsitionfivm stot given that the first player (termed* agent) *plays* a.

For ease of exposition we normalize the payoffs of each state game to be non-negative real numbers 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. The set of possible histories of length t is $(S \times A^2)^t$, and the set of possible histories, /f, 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. Given an SCSG, a policy for the agent is a mapping fromH 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* depends only on *S* instead of on *H.* Such a policy associates with each state a probability distribution on the actions.

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 *p,* 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). Let $U_M(s,\pi,T) = \min_{\rho \text{ is a policy}} U_M(s,\pi,\rho,T)$ denote the value that π can guarantee in T steps starting in 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.

In the sequel 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).

The ergodicity assumption is consistent with the treatment of [Kearns and Singh, 1998], 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 KS's contributions is the identification of the central parameter upon which the analysis of algorithms for learning in MDPs must be based, namely, the *mixing time.* KS 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 *e-neturn 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 policy *n* belongs to the set $\Pi(\epsilon, T)$ of stationary policies whose e-ieturn 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 until our average accumulated reward is sufficiently close to the value of π. Notice an agent that already knows an optimal policy π whose c-return mixing time is T, will need this much time, on the average, to obtain a value of (almost) *v.* Clearly, one cannot expect an agent lacking this information to perform better.

that (1) the probability of failure of learning all columns in at least one set (from among k^2N sets) o$(\frac{2TkNX}{\theta})^3$ r s a r y deviations (i.e. selections of unknown columns) is smaller than k^2N times the probability offailing to learn in one such set of deviations, and that (2) there are at most k^2N entries to learn.

We run the the algorithm for Y stages, such that $\frac{(\frac{2TkNX}{\theta})^3k^2N}{Y} < \frac{\theta}{2}$ 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 $Y > (\frac{2TkNX}{\theta})^3k^2N \cdot \frac{2}{\theta}$. We also need to require that $k^2Ne^{\frac{-2\theta NTX}{\theta}} < \gamma$. Hence, $e^{\frac{2\theta NTX}{\theta}} > \frac{k^2N}{\gamma}$, or $X > \log(\frac{k^2N}{\gamma})\frac{\theta}{2\theta NT}$. 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 $1-\frac{\theta}{2}$ ofthe stages an expected pay off 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 stales (with overwhelming probability). Thus, we trade-off some exploitation for exploration in a manner that guarantees that if the adversary plays an unknown column polynomialy many tunes, 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

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

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 foil owing 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 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 $_{\pi m}$ is executed.[3] The run of $_{\pi m}$ 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 readied 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 l e a$\frac{\epsilon}{p_{max}}$ i t h i n steps.
6. In all cases, whenever an entry in a state game is learned, the value of it is kept in memory.

It is clear (from Lemma 3) that the above algorithm is polynomial (i.e. leads to near optimal average return after polynomial time) in the appropriate parameters. It remains to be shown that this algorithm will yield the desired return with probability of at least $1-\delta$ for a g i v$\delta > 0$.T o show this, we have to consider the four sources of failure of the algorithm. The first three appear in the context of the E^3 algorithm, while the fourth stems from the need to perform local exploration.

1. In some states the algorithm may have a poor estimate of the true next-state distribution. Using standard Chernoff bound analysis [Alon *et al,* 1992], as was applied by KS in the case of MDPs, we can show that, if the number of times an entry was explored is sufficiently large (kit still polynomial), the probability of an error larger than we wish for is small. Notice that, for this analysis, our definition of known states enables us to ignore the fact the rewards in some columns are only partially known.
2. Repeated attempted explorations may fail to expose new information. This can be either because of failure to reach an unknown state and failure to sample a new entry in an unknown state. We can view a global exploration step, followed by random wandering, as a Bernoulli trial, with a constant positive probability of success of at least $O(\frac{\epsilon}{p_{max}k})$ for reaching an unknown state and exploring a new entry in that state. The number of such trials which might be executed before all states become known can therefore be taken (since all of the trials can be treated as independent trials) to be polynomial, with a failure probability of at most $\frac{\delta}{4}$. Notice that in general, not all states need to become known.
3. 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 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 $\frac{\delta}{4}$. This is obtained by standard Chernoff bounds, and

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

makes use of the fact that the standard deviation of the expected reward in a T-step policy is bounded.

4. 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 $\frac{\delta}{4}$ at each of the above stages, we are able to get the desired result

Finally, we remove the assumptions that both the value and its e-return mixing time are known. This is straightforward and almost identical to the treatment given by [Kearns and Singh, 1998]. 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. One can calculate this probability without knowledge of the value and perform exploration whenever this probability exceeds the desired bound. Notice that by employing this technique, with overwhelming probability we remain with known states only after polynomial time. At this point, we can compute an optimal policy. Hence, we can safely apply this exploration bias.

Next, we must deal with the lack of knowledge of T. The idea is as follows: from the proofs 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 a f t$O(P(T_0)^2)$p s , 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 To is the e-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 a restricted class of stochastic games. This algorithm extends earlier work of Kearns mid Singh {1998] on learning in MDPs using the techniques of Monderer and Tennenholtz [1997] for learning in repeated games. These results can be extended to stochastic games in general, as explained in the full paper [Brafman and Tennenholtz, 1998]. Unfortunately, die adversary's ability to influence transitions can lead to very large mixing times and slower, though still polynomial, convergence.

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. It is worth noting a simple, but interesting, corollary of our results. If the algorithm is run concurrently by the agent and the adversary, where we consider stochastic games with stationary equilibrium, they are both guaranteed to attain near optimal performance, i.e., the value of the game (within the allowed error bounds). Finally, we remark that our algorithm would seem a natural candidate for learning in non-stochastic environment, although additional assumptions about the nature of the environment could be used to improve its efficiency.

Acknowledgement: We thank the anonymous reviewers for their useful comments. The first author was partially funded by the Paul Ivanier Center for Robotics Research and Production Management.

References

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

[Brafman and Tennenholtz, 1998] R.I. Brafman and M. Tennenholtz. A near-optimal polynomial time algorithm for learning in stochastic games. Technical Report, Ben-Gurion University, 1998.

[Horijkand Kallenberg, 1981] A. Horijkand L.CM. Kallenberg. Linear Programming and Markov Games. In O. Moeschlin, editor, *Game Theory and Mathematical Economics,* pages 307-319. North Holland, 1981.

[Hu and Wellman, 1998] J. Hu and M.P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proc 15th ICML,* 1998.

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

ttittman, 1994] 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.

[Monderer and Tennenholtz, 1997] O. Monderer and M. Tennenholtz. Dynamic Non-Bayesian Decision-Making. *JAJR,* 7:231-248,1997.

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

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

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