Approximation Results on Sampling Techniques for Zero-sum, Discounted Markov Games

Uday Savagaonkar uday.r.savagaonkar@intel.com
Intel Corporation, Ra1-319
2501 NW 229th Ave
Hillsboro, OR 97124, USA

Robert L. Givan givan@purdue.edu
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285, USA

Edwin K. P. Chong echong@engr.colostate.edu
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, CO 80523, USA

Editor: Leslie Pack Kaelbling

Abstract

We extend the "policy rollout" sampling technique for Markov decision processes to Markov games, and provide an approximation result guaranteeing that the resulting sampling-based policy is closer to the Nash equilibrium than the underlying base policy. This improvement is achieved with an amount of sampling that is independent of the state-space size. We base our approximation result on a more general approximation theorem, proven here, that can be used to analyze other sampling techniques to get similar guarantees. We exhibit this generality by using the theorem to provide an alternate proof of the (already known) result that the "sparse sampling" policy for Markov games approximates the Nash equilibrium. Finally, we provide empirical results showing the utility of the policy rollout technique on both a familiar "soccer game" and a difficult bandwidth-pricing domain. The latter gives an example application of the technique when the assumptions of zero-sum rewards and discounting are relaxed.

Keywords: zero-sum Markov games, sampling techniques, large state-spaces

1. Introduction

Markov Decision Processes (MDPs) (Bellman, 1957, Howard, 1960) provide a powerful framework for modeling situations where a single controller needs to make decisions to achieve a certain goal. Numerous algorithms, such as value iteration and policy iteration, are available for finding (approximately) optimal policies for MDPs (Puterman, 1994, Bertsekas, 1995a, Mine and Osaki, 1970). Algorithms for MDPs have also been developed that can run in times sub-linear in the state-space size, and so are suitable for "large state spaces." These include neuro-dynamic programming by Bertsekas and Tsitsiklis (1996), policy rollout by Tesauro and Galperin (1996) (see also Bertsekas and Castanon, 1999), hindsight optimization by Chong et al. (2000) and Chang (2001), and sparse sampling by Kearns et al. (1999), among many others less related to our work here (e.g, Boutilier et al., 2000, 2001, Yoon et al., 2002, Givan et al., 2003, Fern et al., 2003).

Markov games (Shapley, 1953, Başar and Olsder, 1995) are a natural extension of MDPs to the case where more than one controller is present. The controllers typically have different (possibly conflicting) goals, and thus the usual notion of optimality cannot be extended to Markov games. Instead, the notion of a Nash equilibrium , in which each player's policy is a "best response" to the other players' policies, is a widely accepted notion of optimality in such situations. Much work has been done on the existence and computation of Nash equilibria for the classical finite and infinite games (Başar and Olsder, 1995). But these methods do not generalize to Markov games in a straightforward fashion.

Game theorists have extended value iteration and policy iteration to Markov games under various settings. For example, Patek and Bertsekas (1999) have established the convergence of extensions of these algorithms for stochastic shortest-path games. These offline algorithms typically require time at least polynomial in the state-space cardinality, and are thus not practical for Markov games with large state spaces.

Online algorithms for finding Nash-optimal policies have been developed for some classes of Markov games. Hu and Wellman (1998) have developed an algorithm called Nash-Q that learns the Nash-optimal policy for a restricted class of Markov games. (The class is complex and unnatural, and no membership test is provided.) Littman's friend-or-foe Q-learning is a related method that converges to an equilibrium when it is known that there is a coordination equilibrium or when it is known that there is an adversarial equilibrium (Littman, 2001), and guarantees convergence to a fixed policy in any case.[1] Littman and Szepesvári (1996) have also provided a more generalized framework for solving such problems. Bowling and Veloso (2002) have developed a variable-rate gradient-based learning algorithm for two-player, two-action repeated matrix games. But none of these algorithms can be used for Markov games with large state spaces.

Difficulties applying such algorithms in the case of large state spaces are not unique to Markov games. Similar problems exist for MDPs as well. The algorithms that researchers have developed to tackle such problems mainly fall in one of three categories—structure exploitation, value-function approximation, and sampling.

In this paper, we focus on sampling algorithms. The sampling algorithms we consider involve drawing random samples to estimate, for each possible initial action pair, the value of taking that initial action pair, and then either acting optimally, or following some given policy pair. We call this estimating the Q-function (for the policy pair, if any). The resulting Q-function estimate defines a matrix game for the current state, and a current action is then chosen (possibly stochastically) by finding a (possibly mixed) Nash equilibrium for this game. Our aim is to evaluate policies that are formed using this "Nash look-ahead" technique. In this evaluation, one needs to analyze how the error in the estimated Q-function propagates to the value function of the resulting policy, as well as the significance of the policy pair chosen, if any.

Kearns et al. (1999) have given a "sparse sampling" method for drawing the random samples just described to estimate the optimal Q-function and produce a near-Nash stochas-

1. In a coordination equilibrium, all players attain their maximal value, and in an adversarial equilibrium, each player benefits from any change by the other players.

tic policy. Here, we focus instead on estimating the Q-function for particular policy pairs, in order to find an "improved" policy for one player.[2] This work is a generalization of the "policy rollout" technique for MDPs to Markov games, and has not previously been analyzed. The motive for considering policy rollout in place of directly approximating the optimal Q-function is the exponential cost associated with the sparse sampling method (relative to the desired bound on loss).

The resulting policy-rollout method for Markov games is an online version of policy iteration for Markov games, much as policy rollout provides an online version of policy iteration for MDPs. Here, we provide the details of this method and elucidate the conditions under which policy improvement is guaranteed. We also show that the amount of sampling required to guarantee policy improvement is independent of the size of the state space—this result is new even for policy rollout in MDPs.

Along the way, we prove an approximation result for discounted zero-sum games that provides bounds on the loss of the Nash look-ahead policy constructed using a sampled Q-function estimation. This result naturally implies our desired approximation guarantees, but is general enough to be useful in analyzing other sampling methods as well. As an illustration, we provide a brief alternate proof of the (already known) near-Nash behavior of sparse sampling for Markov games based on this theorem. Since MDPs are special cases of Markov games, this result also easily implies the approximation guarantees previously shown for both policy rollout and sparse sampling in MDPs.

Weaker, similar, related approximation results have been shown previously by Singh and Yee (1994) for MDPs and by Lagoudakis and Parr (2002) for Markov games. We use different techniques than those used to prove these results, and we discuss the ways in which our approximation result extends these previous results in Section 3.2. In particular, we consider approximation of the Q-functions of arbitrary base-policies, as in rollout, not just the optimal Q function; we incorporate the uncertainty introduced by sampling; and we provide bounds on the loss at individual states relative to their loss in the base policy which can be tighter than the sup-norm bounds that apply to the entire policy—these are of particular interest in analyzing rollout algorithms, where the base policy may vary in quality across the state space.

Our analysis is similar to that establishing the convergence of policy iteration for Markov games, by Patek and Bertsekas (1999)—this similarity results because of the close relationship between policy rollout and policy iteration, in both MDPs and Markov games. However, preserving that previous result under finite sampling and finite horizon length (to make the algorithm implementable online) requires a non-trivial extension of that work. For this extension, we develop and use the new approximation theorem mentioned above.

We also exhibit the performance of policy rollout empirically in two domains, showing that policy improvement can be achieved with a practical amount of sampling in each. We consider the zero-sum "soccer game" introduced by Littman (1994) and then the much more complex, general-sum bandwidth-pricing domain of Savagaonkar (2002). Although our theorem guarantees policy improvement only for zero-sum games, we show empirical policy improvement even in the latter substantial general-sum game.

2. "Improved" here means closer to Nash-optimal.

2. Definitions, Notation, and Technical Background

Throughout this paper, for any set $\mathbb{S}$, we use $\Pi(\mathbb{S})$ to denote the space of all the probability measures over $\mathbb{S}$. We use bold-face fonts to indicate random variables. Given two real-valued functions f and g on the same domain $\mathbb{D}$, we write $f \leq g$ to indicate that $f(x) \leq g(x)$ for every x in $\mathbb{D}$. We write $|f|_\infty$ for $\sup_{x \in \mathbb{D}} |f(x)|$.

A zero-sum, discounted Markov game between players A and B with a discount factor γ is a six-tuple $\langle \mathbb{X}, \mathbb{A}, \mathbb{B}, \mathfrak{T}, R, x_0 \rangle$, where $\mathbb{X}$ is the (countable) state space, $\mathbb{A}$ ($\mathbb{B}$) is the action space for player A (B), $\mathfrak{T} : \mathbb{X} \times \mathbb{A} \times \mathbb{B} \to \Pi(\mathbb{X})$ is the transition function, $R : \mathbb{X} \times \mathbb{A} \times \mathbb{B} \to \mathbb{R}$ is the reward function, and $x_0 \in \mathbb{X}$ is the initial state. The aim of A is to maximize the discounted reward (with discount factor γ) defined using R, and that of B is to minimize the same. We assume that the action spaces $\mathbb{A}$ and $\mathbb{B}$ have finite cardinality. For notational convenience, we will denote by $\boldsymbol{f}(x, \boldsymbol{a}, \boldsymbol{b})$ a random state resulting from players A and B taking actions chosen from $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively, in state x, as specified by $\mathfrak{T}$.

A policy π^A for player A is a sequence of maps $\langle \mu_0^{\pi^A}, \mu_1^{\pi^A}, \cdots \rangle$, where each $\mu_i^{\pi^A} : \mathbb{X} \to \Pi(\mathbb{A})$ specifies the probability distribution with which actions are chosen by A in each of the states at time i. If $\mu_i^{\pi^A} = \mu_0^{\pi^A}$ for all i, then the policy is said to be stationary. Similarly, a policy π^B for player B is a sequence of maps $\langle \mu_0^{\pi^B}, \mu_1^{\pi^B}, \cdots \rangle$, with each $\mu_i^{\pi^B} : \mathbb{X} \to \Pi(\mathbb{B})$, and again, if $\mu_i^{\pi^B} = \mu_0^{\pi^B}$ for all i, then the policy is said to be stationary. For notational convenience, given a map $\mu^A : \mathbb{X} \to \Pi(\mathbb{A})$ and a state $x \in \mathbb{X}$, we use the bold-face notation $\boldsymbol{\mu}^A(x)$ to denote a random variable that has distribution $\mu^A(x)$, and we similarly define $\boldsymbol{\mu}^B(x)$ for map $\mu^B : \mathbb{X} \to \Pi(\mathbb{B})$. We sometimes refer to policies for player A as A-policies and those for B as B-policies.

Given a policy π for either player, we use the notation μ_k^π to denote the k'th member of the sequence π. When the policy π is stationary, we will omit the subscript k. Given a pair of policies $\langle \pi^A, \pi^B \rangle$, we define the value of the game in state x as

$$V_{\pi^A, \pi^B}(x) = E\left[\sum_{k=0}^{\infty} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\pi^B}(\boldsymbol{x}_k))\right], \tag{1}$$

where $\boldsymbol{x}_0 = x$, and $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\pi^B}(\boldsymbol{x}_k))$. The function $V_{\pi^A, \pi^B} : \mathbb{X} \to \mathbb{R}$ is called the value function corresponding to the pair $\langle \pi^A, \pi^B \rangle$. Generally, we refer to any $V : \mathbb{X} \to \mathbb{R}$ as a value function. The space of all value functions is denoted by $\mathbb{V}$.

Given an A-policy π^A, a corresponding best-response policy[3] for B is defined as a B-policy π^B that minimizes the value $V_{\pi^A, \pi^B}(x)$ of the game in each state x, given that A plays policy π^A. There could be more than one best-response policy, but the value achieved at each state does not vary among the best-response policies. We denote the set of all the best-response policies for B by $\mathrm{brB}(\pi^A)$. Similarly, we write $\mathrm{brA}(\pi^B)$ for the set of best-response policies for A to a given policy π^B for B, each of which maximizes the value of the game, given that B plays π^B. A pair of policies $\langle \pi^A, \pi^B \rangle$ is said to constitute a Nash equilibrium, if $\pi^B \in \mathrm{brB}(\pi^A)$ and $\pi^A \in \mathrm{brA}(\pi^B)$, in which case we call both policies *Nash policies* for the respective players.

3. Given a fixed policy for A, the game reduces to an MDP from B's perspective, and it follows from basic MDP theory that a best-response policy exists.

For any A-policy π^A and B-policy π^B, define the security levels $V^s_{\pi^A}$ and $V^s_{\pi^B}$ as

$$V^s_{\pi^A}(x) \triangleq \min_{\tilde{\pi}^B} V_{\pi^A,\tilde{\pi}^B}(x) \quad \text{and} \quad V^s_{\pi^B}(x) \triangleq \max_{\tilde{\pi}^A} V_{\tilde{\pi}^A,\pi^B}(x).$$

Note that if $\pi^B \in \mathrm{brB}(\pi^A)$, then $V^s_{\pi^A} = V_{\pi^A,\pi^B}$, and if $\pi^A \in \mathrm{brA}(\pi^B)$ then $V^s_{\pi^B} = V_{\pi^A,\pi^B}$. The policy pair $\langle \pi^A, \pi^B \rangle$ is a Nash equilibrium when $V^s_{\pi^A} = V^s_{\pi^B}$. We denote this value function by V^*. In general, $V^s_{\pi^A} \leq V^*$ and $V^s_{\pi^B} \geq V^*$. At any state x, we say the A-policy π^A incurs a *loss* of $|V^*(x) - V^s_{\pi^A}(x)|$, and likewise for B-policies π^B.

The notion of security level allows a natural way of comparing two policies. For MDPs, ordering the policies is relatively easy, as we can compare the value functions directly. But for Markov games, existence of the opponent complicates the situation, as no assumption can be made about the opponent policy. But the security level of a policy is the worst-case performance of that policy, and thus is independent of the opponent. Hence, a partial ordering on policies can be introduced using security levels. Given $x \in \mathbb{X}$, we say that A-policy $\tilde{\pi}^A$ is better than A-policy π^A in state x, if $V^s_{\tilde{\pi}^A}(x) \geq V^s_{\pi^A}(x)$. If this holds in every state, then we say that $\tilde{\pi}^A$ is state-wise better π^A. In policy improvement algorithms, state-wise improvement is desirable, but is not always easy to obtain. Hence, we use a weaker notion of policy improvement.

Definition 1 *We say that policy $\tilde{\pi}^A$ is better than policy π^A in the sup norm if* $\left|V^* - V^s_{\tilde{\pi}^A}\right|_\infty \leq \left|V^* - V^s_{\pi^A}\right|_\infty$.

Let $V : \mathbb{X} \to \mathbb{R}$ be a value function. Define the operator T as

$$(TV)(x) = \max_{z \in \Pi(\mathbb{A})} \min_{b \in \mathbb{B}} E\left[R(x, \boldsymbol{z}, b) + \gamma V(\boldsymbol{f}(x, \boldsymbol{z}, b))\right]. \tag{2}$$

Then $(T^k V)(x)$ is the k-stage optimal discounted reward for player A, with starting state of x and terminal reward of V. Also for pair of policies $\langle \pi_A, \pi_B \rangle$, define

$$(T_{\mu_k^{\pi^A},\mu_k^{\pi^B}} V)(x) = E\left[R(x, \boldsymbol{\mu}_k^{\pi^A}(x), \boldsymbol{\mu}_k^{\pi^B}(x)) + \gamma V(\boldsymbol{f}(x, \boldsymbol{\mu}_k^{\pi^A}(x), \boldsymbol{\mu}_k^{\pi^B}(x)))\right]. \tag{3}$$

We abuse notation by using $T^k_{\pi^A,\pi^B}$ to denote something other than k iterative applications of an operator, as follows:

$$\begin{aligned}
(T^0_{\pi^A,\pi^B} V)(x) &= V(x), \\
(T^k_{\pi^A,\pi^B} V)(x) &= (T^{k-1}_{\pi^A,\pi^B}(T_{\mu_k^{\pi^A},\mu_k^{\pi^B}} V))(x).
\end{aligned}$$

Unless π^A and π^B are stationary, the operator being applied at each iteration changes. $(T^k_{\pi^A,\pi^B} V)(x)$ is the k-stage discounted reward for player A when the players use the policy pair $\langle \pi^A, \pi^B \rangle$ and the terminal reward is V. For a stationary policy pair $\langle \pi^A, \pi^B \rangle$, we will use the short-hand notation $T_{\pi^A,\pi^B} \triangleq T^1_{\pi^A,\pi^B}$.

The Q-function $Q_{\pi^A \pi^B} : \mathbb{X} \times \mathbb{A} \times \mathbb{B} \to \mathbb{R}$ for stationary $\langle \pi^A, \pi^B \rangle$, is defined as

$$Q_{\pi^A,\pi^B}(x, a, b) = E\left[R(x, a, b) + \gamma V_{\pi^A,\pi^B}(\boldsymbol{f}(x, a, b))\right].$$

In general, we will call any function $Q : \mathbb{X} \times \mathbb{A} \times \mathbb{B} \to \mathbb{R}$ a Q-function. We write $\mathbb{Q}$ for the space of all the Q-functions. Also, if $\langle \pi^A, \pi^B \rangle$ constitute a Nash equilibrium, then we write $Q^*(\cdot,\cdot,\cdot)$ for $Q_{\pi^A,\pi^B}(\cdot,\cdot,\cdot)$. Note that given a state $x \in \mathbb{X}$ and a function $Q \in \mathbb{Q}$, $Q(x,\cdot,\cdot)$ is a matrix of dimension $|\mathbb{A}| \times |\mathbb{B}|$. We are interested in estimating Q-functions stochastically, so we consider random variables that take values in $\mathbb{Q}$. We call such random variables *stochastic Q-functions*. Here, the distribution for a stochastic Q-function will often be specified by a sampling algorithm, and may not be described in a closed form. The algorithm itself represents the distribution implicitly.

Let $\mathrm{Nash}(M(\cdot,\cdot))$ be an operator that for any matrix $M(\cdot,\cdot) \in \mathbb{R}^{|\mathbb{A}|\times|\mathbb{B}|}$ computes a probability distribution pair $\langle \mu^A, \mu^B \rangle$ that achieves a Nash equilibrium for the zero-sum matrix game[4] described by $M(\cdot,\cdot)$. Here, $\mu^A \in \arg\max_{z\in\Pi(\mathbb{A})} \min_{b\in\mathbb{B}} E\left[M(\boldsymbol{z}, b)\right]$, and $\mu^B \in \arg\min_{z\in\Pi(\mathbb{B})} \max_{a\in\mathbb{A}} E\left[M(a, \boldsymbol{z})\right]$. Our results are independent of which such operator is selected here (i.e., which Nash equilibrium is returned for each matrix). The operator $\mathrm{NashVal}(M(\cdot,\cdot))$ returns the value $\max_{z\in\Pi(\mathbb{A})} \min_{b\in\mathbb{B}} E\left[M(\boldsymbol{z}, b)\right]$ of the matrix game $M(\cdot,\cdot)$ when the $\mathrm{Nash}(M(\cdot,\cdot))$ distributions are used by the players. Note that the operator $\mathrm{Nash}(M(\cdot,\cdot))$ can be implemented as a linear program (Başar and Olsder, 1995, Bertsekas, 1995b) and the operator $\mathrm{NashVal}(M(\cdot,\cdot))$ can be implemented as simple matrix multiplication (Başar and Olsder, 1995), in addition to the same linear program. For notational convenience, we also define the operators $\mathrm{Nash}_A(M)$ and $\mathrm{Nash}_B(M)$ to return the player A and player B components of $\mathrm{Nash}(M)$, respectively.

Because of our interest in sampling, the matrix M describing the matrix game to be solved will often be itself a random variable $\boldsymbol{M}$. In this case, $\mathrm{NashVal}(\boldsymbol{M})$ is a function of this random variable. Standard probability theory implies that $\mathrm{NashVal}(\boldsymbol{M})$ can be written then as $\max_{z\in\Pi(\mathbb{A})} \min_{b\in\mathbb{B}} E\left[\boldsymbol{M}(\boldsymbol{z}, b)|\boldsymbol{M}\right]$.

Throughout this paper, we assume that the reward function is bounded, i.e., $|R(x,a,b)| \leq R_{\max}$ for some $R_{\max} \in \mathbb{R}$ and all $x \in \mathbb{X}$, $a \in \mathbb{A}$, and $b \in \mathbb{B}$. This immediately implies that the value function for any policy pair $\langle \pi^A, \pi^B \rangle$ satisfies, $|V_{\pi^A,\pi^B}(x)| \leq V_{\max}$ for all $x \in \mathbb{X}$, where $V_{\max}$ is defined as $V_{\max} \triangleq R_{\max}/(1-\gamma)$. We also use the special symbol e to represent a value function $e : \mathbb{X} \to \mathbb{R}$ such that $e(x) = 1$ at every state $x \in \mathbb{X}$.

The following technical background propositions, simply extending results for MDPs presented by Bertsekas (1995a) and (for the last proposition) Alon et al. (1992), are proven in Appendix A, for completeness. Here, let $V(\cdot)$ and $V'(\cdot)$ be value functions, and $\langle \pi^A, \pi^B \rangle$ be a policy pair.

Proposition 2 *Suppose $V(x) \leq V'(x)$, for all $x \in \mathbb{X}$. Then, for all $x \in \mathbb{X}$,*

$$\begin{aligned} (T^k V)(x) &\leq (T^k V')(x), \quad \textit{and} \\ (T^k_{\pi^A,\pi^B} V)(x) &\leq (T^k_{\pi^A,\pi^B} V')(x). \end{aligned}$$

Proposition 3 *For any $r \in \mathbb{R}$, and e the unit value function,*

$$\begin{aligned} (T^K(V + re))(x) &= (T^K V)(x) + \gamma^K r, \quad \textit{and} \\ (T^K_{\pi^A,\pi^B}(V + re))(x) &= (T^K_{\pi^A,\pi^B} V)(x) + \gamma^K r. \end{aligned}$$

4. Matrix games are well-described elsewhere, e.g., by Başar and Olsder (1995). A matrix game can be viewed as a Markov game with one state and reward function given by the matrix.

Proposition 4 $\sup_{\pi^A} \inf_{\pi^B} (T^K_{\pi^A,\pi^B} V)(x) = (T^K V)(x)$.

Proposition 5 (*Value iteration converges*) $\lim_{N\to\infty} (T^N V)(x_0) = V^*(x_0)$.

Proposition 6 $\lim_{N\to\infty} (T^N_{\pi^A,\pi^B} V)(x_0) = V_{\pi^A,\pi^B}(x_0)$.

Proposition 7 *For stationary π^A and π^B, $V_{\pi^A,\pi^B} = T_{\pi^A,\pi^B} V_{\pi^A,\pi^B}$.*

Proposition 8 *The Nash value satisfies Bellman's equation, $V^* = TV^*$.*

Proposition 9 *Suppose V and V' are bounded. For all $k \in \mathbb{N}$, we have*

$$\max_{x\in\mathbb{X}} \left| (T^k V)(x) - (T^k V')(x) \right| \leq \gamma^k \max_{x\in\mathbb{X}} \left| V(x) - V'(x) \right| .$$

Proposition 10 *Let $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_N$ be i.i.d. random variables satisfying $|\boldsymbol{Y}_i| \leq Y_{\max}$ w.p.1 and $E\boldsymbol{Y}_i = \mu$. Then, $P\left[\left| \frac{1}{N} \sum_{i=1}^N \boldsymbol{Y}_i - \mu \right| \leq \lambda \right] \geq 1 - 4e^{-\lambda^2 N/(8Y_{\max}^2)}$.*

3. An Approximation Result for Markov Games

3.1 The Concept of Look-ahead

Policy selection in MDP problems often involves a one-step look-ahead relative to a given value function, selecting an action that will maximize the value if the future value is given by the specified value function. The corresponding process in Markov games is more complicated.

One step look-ahead can convert a given value function into a Q-function; however, maximization is insufficient for action selection, as the opponent action is unknown. Our approach to analyzing Markov games leverages the idea that the Q function defines a matrix game in the current state. This matrix game can be solved to get a "Nash-equilibrium action distribution", analogous to the best action in the MDP case. This idea appears in work on policy iteration for Markov games by Patek and Bertsekas (1999) and underlies our approach. Here, we must be concerned with the effect of sampling error on the resulting policy improvement—because of sampling, we do not have a Q-function, but rather a stochastic Q-function. Analysis of the effects of sampling is critical for large state-space games, where exact methods are not practical. Below, our look-ahead function converts a given distribution F over Q-functions (typically, from a sampling algorithm) into a distribution over actions.

Definition 11 *Given a distribution F over $\mathbb{Q}$, the one-step look-ahead policy* $\mathrm{lookahead_A}(F)$ *for A chooses an action in state x according to the probability distribution E* $\mathrm{Nash}_A(\boldsymbol{Q}(x,\cdot,\cdot))$, *where $\boldsymbol{Q}$ is a random variable with distribution F.*

The expectation in Definition 11 is an expectation in the space of probability distributions, computing an expected "Nash-equilibrium" action distribution. The stochastically described matrix $\boldsymbol{Q}(x,\cdot,\cdot)$ can be viewed as a matrix-game encapsulation of the expected future following each available action pair, and the $\mathrm{lookahead_A}(F)$ policy chooses its actions by solving this game. Now suppose that we have a (sampling?) algorithm that takes as an input the current state x, and outputs a random matrix $\boldsymbol{M} \in \mathbb{R}^{|\mathbb{A}| \times |\mathbb{B}|}$ distributed as specified by F in state x. Then the policy $\mathrm{lookahead}_{\mathrm{A}}(F)$ can be generated as follows. At every decision epoch, observe the current state x and generate a random matrix $\boldsymbol{M}$ using the given algorithm, and compute the distribution $\mathrm{Nash}_A(\boldsymbol{M})$, and choose an action a according this distribution. The sampling algorithms we consider use this technique for generating policies.

3.2 The Look-ahead Approximation Theorem

Our main theorem provides bounds on the state-wise and sup-norm loss resulting when following a policy selected by Nash look-ahead using sampled Q-functions. We use this theorem in Sections 4 and 5 to provide bounds on the sampling error (i.e., ϵ and δ) to ensure policy improvement for two different sampling algorithms. The theorem requires that the sampled Q-function approximates Q_{π^A,π^B} for some policy pair $\langle \pi^A, \pi^B \rangle$, where π^B is a best-response to π^A. (When $\langle \pi^A, \pi^B \rangle$ is a Nash equilibrium, the bounds then limit the loss relative to V^*.) Intuitively, the resulting look-ahead policy will have a security level that is (approximately) no worse at any state than that of π^A and, with enough sampling, that is better at the state furthest from its Nash value, contracting to the Nash-equilibrium value with a rate at least equal to the discount factor. We compare our result to related previous results after the proof below.

Theorem 12 *Let π^A be a stationary policy for A, and let π^B be a best-response policy for π^A. Let F be a Q-function distribution such that $\boldsymbol{Q}$ distributed according to F satisfies $\left|Q_{\pi^A,\pi^B}(x,\cdot,\cdot) - \boldsymbol{Q}(x,\cdot,\cdot)\right|_\infty < \epsilon$, for any $x \in \mathbb{X}$, with probability at least $1-\delta$ and is a.s. bounded by $V_{\max}$, i.e., $P[|\boldsymbol{Q}|_\infty \le V_{\max}] = 1$. Let $\tilde{\pi}^A = \mathrm{lookahead}_{\mathrm{A}}(F)$. Then, we have*

$$V^s_{\pi^A}(x) < V^s_{\tilde{\pi}^A}(x) + \frac{2(\epsilon + 2\delta V_{\max})}{(1-\gamma)},$$

for all $x \in \mathbb{X}$. Moreover, for small enough ϵ and δ, there is contraction towards the equilibrium value:

$$|V^* - V^s_{\tilde{\pi}^A}|_\infty < \gamma |V^* - V^s_{\pi^A}|_\infty + \frac{2(\epsilon + 2\delta V_{\max})}{(1-\gamma)}.$$

Proof. Let $\tilde{\pi}^B \in \mathrm{brB}(\tilde{\pi}^A)$ be a stationary best-response policy to $\tilde{\pi}^A$. Let $\boldsymbol{Q}$ be the stochastic Q-function having distribution F. We wish to compare the security level of π^A, i.e., V_{π^A,π^B}, with the security level achieved by $\tilde{\pi}^A$, i.e., $V_{\tilde{\pi}^A,\tilde{\pi}^B}$, and show that the latter approximately dominates the former. To do so, we define an "approximately" increasing sequence of "approximately" intermediate value functions, starting with the expected value of the "look-ahead game" described by the $\boldsymbol{Q}(x,\cdot,\cdot)$ matrix, and ending at the security level of $\tilde{\pi}^A$, as follows:

$$V_1(x) = E\,\mathrm{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) \;\text{ and }\; V_{K+1} = T_{\tilde{\pi}^A,\tilde{\pi}^B} V_K.$$

This sequence necessarily converges to the security level of $\tilde{\pi}^A$—it remains to show that the sequence is approximately increasing, and that V_1 approximately dominates the security level of π^A, analyzing how the approximation bounds sum over the sequence. We turn to the latter first.

For notational conciseness, let us denote the event $|\boldsymbol{Q}(x,\cdot,\cdot) - Q_{\pi^A,\pi^B}(x,\cdot,\cdot)|_\infty < \epsilon$ by $\mathfrak{E}_x$. Then, by our choice of F, we have $P\left[\mathfrak{E}_x\right] > 1-\delta$. Also, we denote the complement of this event by $\mathfrak{E}_x^c$. Thus, $P[\mathfrak{E}_x^c] < \delta$. Now,

$$\begin{aligned}
V_1(x) &= E\,\mathrm{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) = E\left[\max_{\mu^A\in\Pi(\mathbb{A})}\min_{b\in\mathbb{B}} E\left[\boldsymbol{Q}(x,\boldsymbol{\mu}^A,b)\middle|\boldsymbol{Q}\right]\right] \\
&\geq E\left[\min_{b\in\mathbb{B}} E\left[\boldsymbol{Q}(x,\boldsymbol{\mu}^{\pi^A}(x),b)\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x\right] P\left[\mathfrak{E}_x\right] \\
&\quad + E\left[\min_{b\in\mathbb{B}} E\left[\boldsymbol{Q}(x,\boldsymbol{\mu}^{\pi^A}(x),b)\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x^c\right] P\left[\mathfrak{E}_x^c\right] \\
&\geq E\left[\min_{b\in\mathbb{B}} E\left[\boldsymbol{Q}(x,\boldsymbol{\mu}^{\pi^A}(x),b)\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x\right] (1-P\left[\mathfrak{E}_x^c\right]) - P\left[\mathfrak{E}_x^c\right] V_{\max} \\
&> E\left[\min_{b\in\mathbb{B}} E\left[Q_{\pi^A,\pi^B}(x,\boldsymbol{\mu}^{\pi^A}(x),b)-\epsilon\right]\right] - 2\delta V_{\max} \\
&= V^s_{\pi^A}(x) - \epsilon - 2\delta V_{\max}.
\end{aligned}$$

We now show that the sequence of value functions is approximately increasing, starting with the preliminary observation that, when $|\boldsymbol{Q}(x,\cdot,\cdot) - Q_{\pi^A,\pi^B}(x,\cdot,\cdot)|_\infty < \epsilon$ (i.e., the event $\mathfrak{E}_x$ occurs), we have, for any $x\in\mathbb{X}$, $a\in\mathbb{A}$, and $b\in\mathbb{B}$,

$$\begin{aligned}
E\left[R(x,a,b)+\gamma V_1(\boldsymbol{f}(x,a,b))\right] &> E\left[R(x,a,b)+\gamma V^s_{\pi^A}(\boldsymbol{f}(x,a,b)) - \gamma(\epsilon+2\delta V_{\max})\right] \\
&= Q_{\pi^A,\pi^B}(x,a,b) - \gamma(\epsilon+2\delta V_{\max}) \\
&> \boldsymbol{Q}(x,a,b) - \epsilon - \gamma(\epsilon+2\delta V_{\max}).
\end{aligned}$$

Now, writing $\nu(\boldsymbol{Q},x)$ for $\mathrm{Nash}_A(\boldsymbol{Q}(x,\cdot,\cdot))$, and exploiting the definition of $\mu^{\tilde{\pi}^A}(x)$ as $E\,\mathrm{Nash}_A(\boldsymbol{Q}(x,\cdot,\cdot))$,

$$\begin{aligned}
V_2(x) &= (T_{\tilde{\pi}^A,\tilde{\pi}^B} V_1)(x) \\
&= E\left[R(x, \boldsymbol{\mu}^{\tilde{\pi}^A}(x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)) + \gamma V_1(\boldsymbol{f}(x, \boldsymbol{\mu}^{\tilde{\pi}^A}(x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)))\right] \\
&= E\left[E\left[R(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)) + \gamma V_1(\boldsymbol{f}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)))\middle|\boldsymbol{Q}\right]\right] \\
&= E\left[E\left[R(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)) + \gamma V_1(\boldsymbol{f}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)))\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x\right] P\left[\mathfrak{E}_x\right] \\
&\quad + E\left[E\left[R(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)) + \gamma V_1(\boldsymbol{f}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)))\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x^c\right] P\left[\mathfrak{E}_x^c\right] \\
&\geq E\left[E\left[R(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)) + \gamma V_1(\boldsymbol{f}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x)))\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x\right] P\left[\mathfrak{E}_x\right] \\
&\quad - P\left[\mathfrak{E}_x^c\right] V_{\max} \\
&> E\left[E\left[\boldsymbol{Q}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x))\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x\right] P\left[\mathfrak{E}_x\right] - \epsilon - \gamma(\epsilon + 2\delta V_{\max}) \\
&\quad - P\left[\mathfrak{E}_x^c\right] V_{\max} \\
&= E\left[E\left[\boldsymbol{Q}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x))\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x\right] P\left[\mathfrak{E}_x\right] \\
&\quad + (1-1)E\left[E\left[\boldsymbol{Q}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x))\middle|\boldsymbol{Q}\right]\middle|\mathfrak{E}_x^c\right] P\left[\mathfrak{E}_x^c\right] \\
&\quad - \epsilon - \gamma(\epsilon + 2\delta V_{\max}) - P\left[\mathfrak{E}_x^c\right] V_{\max} \\
&\geq E\left[E\left[\boldsymbol{Q}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^{\tilde{\pi}^B}(x))\middle|\boldsymbol{Q}\right]\right] - P\left[\mathfrak{E}_x^c\right] V_{\max} - \epsilon - \gamma(\epsilon + 2\delta V_{\max}) \\
&\quad - P\left[\mathfrak{E}_x^c\right] V_{\max} \\
&\geq E\left[\min_{\mu^B} E\left[\boldsymbol{Q}(x, \boldsymbol{\nu}(\boldsymbol{Q}, x), \boldsymbol{\mu}^B(x))\middle|\boldsymbol{Q}\right]\right] - \epsilon - \gamma(\epsilon + 2\delta V_{\max}) - 2\delta V_{\max} \\
&= V_1(x) - (1+\gamma)(\epsilon + 2\delta V_{\max}), \text{ by the definitions of } \nu() \text{ and } V_1.
\end{aligned}$$

Using this fact, Propositions 2 and 3 imply that for all $K \geq 1$, $V_{K+1}(x) \geq V_K(x) - \gamma^{k-1}(1+\gamma)(\epsilon + 2\delta V_{\max})$. Also, as shown above, $V_1(x) > V^s_{\pi^A} - \epsilon - 2\delta V_{\max}$. Then, by Proposition 6, we have

$$\begin{aligned}
V^s_{\tilde{\pi}^A}(x) = \lim_{K\to\infty} V_K(x) &\geq V_1(x) - \left(\sum_{k=0}^{\infty} \gamma^k (1+\gamma)(\epsilon + 2\delta V_{\max})\right) \\
&= V_1(x) - (1+\gamma)(\epsilon + 2\delta V_{\max})/(1-\gamma) \\
&> V^s_{\pi^A}(x) - (\epsilon + 2\delta V_{\max}) - (1+\gamma)(\epsilon + 2\delta V_{\max})/(1-\gamma) \\
&= V^s_{\pi^A}(x) - 2(\epsilon + 2\delta V_{\max})/(1-\gamma).
\end{aligned}$$

We have now shown the first claim in the theorem, and turn to bounding the loss in the sup norm, using the tools just developed. Let us define a new value function $V_1' = TV^s_{\pi^A}$. Then, for each state x, we have $V_1'(x) = \mathrm{NashVal}(Q_{\pi^A,\pi^B}(x,\cdot,\cdot))$. Now, when we have $|\boldsymbol{Q}(x,\cdot,\cdot) - Q_{\pi^A,\pi^B}(x,\cdot,\cdot)|_\infty < \epsilon$ (i.e., the event $\mathfrak{E}_x$ occurs), we must also have $|\,\mathrm{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) -$

$\text{NashVal}(Q_{\pi^A,\pi^B}(x,\cdot,\cdot))| < \epsilon$. Thus,

$$\begin{aligned}
|V_1(x) - V_1'(x)| &= |E\,\text{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) - \text{NashVal}(Q_{\pi^A,\pi^B}(x,\cdot,\cdot))| \\
&\leq E|\,\text{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) - \text{NashVal}(Q_{\pi^A,\pi^B}(x,\cdot,\cdot))| \\
&= E\left[|\,\text{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) - \text{NashVal}(Q_{\pi^A,\pi^B}(x,\cdot,\cdot))|\,\middle|\,\mathfrak{E}_x\right] P\left[\mathfrak{E}_x\right] \\
&\quad + E\left[|\,\text{NashVal}(\boldsymbol{Q}(x,\cdot,\cdot)) - \text{NashVal}(Q_{\pi^A,\pi^B}(x,\cdot,\cdot))|\,\middle|\,\mathfrak{E}_x^c\right] P\left[\mathfrak{E}_x^c\right] \\
&< \epsilon + 2\delta V_{\max}.
\end{aligned}$$

But then, using the lower bound shown for $V^s_{\tilde{\pi}^A}$ on page 10,

$$V^*(x) \geq V^s_{\tilde{\pi}^A}(x) \geq V_1(x) - \frac{(1+\gamma)(\epsilon + 2\delta V_{\max})}{1-\gamma} > V_1'(x) - \frac{2(\epsilon + 2\delta V_{\max})}{1-\gamma}.$$

This when combined with Proposition 9 and the definition of V_1' as $TV^s_{\pi^A}$ gives

$$|V^* - V^s_{\tilde{\pi}^A}|_\infty < |V^* - V_1'|_\infty + \frac{2(\epsilon + 2\delta V_{\max})}{(1-\gamma)} \leq \gamma|V^* - V^s_{\pi^A}|_\infty + \frac{2(\epsilon + 2\delta V_{\max})}{(1-\gamma)}.$$

■

This result is for discounted Markov games, and is more general[5] than that proven by Singh and Yee (1994) in the following four respects.

1. We extend the MDP result to Markov games.
2. We relax the finite–state-space restriction to allow countable state-spaces.[6]
3. The previous result is stated only for the case where the value function (Q-function) being approximated is the optimal value function (optimal Q-function) for the problem. Our result applies to approximation of security level (or Q-function) for an arbitrary "base policy". This generality is of particular interest in analyzing rollout.
4. We bound both the sup-norm loss for the look-ahead policy and the loss suffered by each state relative to the loss for that state in the base policy. This extension is of particular interest in analyzing rollout, where the base policy may be quite good in some states, so that the sup-norm bound (which derives from the performance at the worst state) is very loose at those states.

Because of items 3 and 4 in this list, we used a different proof method from Singh and Yee. Here, we non-trivially extended the techniques of Patek and Bertsekas (1999) to include bounds on the effects of Q-function approximation.

Another result of interest is that by Lagoudakis and Parr (2002), providing bounds on the performance of a policy obtained by using an approximation of the optimal Q-function for look-ahead. Our bounds are more general than this previous work in the following ways.

5. Results proven for Markov games immediately imply those for MDPs, as an MDP can be looked at as a Markov game, where the opponent's action space has only one element
6. We believe that, with suitable regularity assumptions, this result extends immediately to continuous state spaces, but we do not explore that claim further here.

1. The previous result is stated only for approximations of the optimal Q-functions. Our result applies to approximations of Q-functions of general policies.
2. The previous analysis does not incorporate the uncertainty introduced by sampling.
3. The previous analysis places bounds on the sup norm of the loss resulting from the approximation. Our result, in addition to bounding the sup-norm loss, also bounds the loss at individual states, relative to that suffered by the base policy—this bound can be tighter than the sup-norm bound for states where the base policy performs well.
4. The previous bounds are stated in terms of the *residual*[7] of the approximate value. It is not clear how to compute the residual of a sampled value function without performing the max/min operation across the state space, which is expensive here. Bounding the residual of V loosely with $(V - V^*)/(1-\gamma)$ and using the previous result gives a looser guarantee than that provided by our Theorem 12. Our bound, in contrast, is based on ϵ-δ probabilistic guarantees on the accuracy of V that can be guaranteed by sufficient sampling.

4. Policy Rollout for Markov Games

4.1 The Algorithm

Policy rollout is a recently developed technique used for policy improvement in MDPs (Tesauro and Galperin, 1996, Bertsekas and Castanon, 1999). The technique starts with a base policy, and uses sampling to determine the Q-function of that policy. It then uses this Q-function for one-step look-ahead to choose optimal actions. Such a policy is shown by Bertsekas and Castanon (1999) to be no worse than the base policy for a wide class of MDPs.

Here, we extend the policy rollout technique to zero-sum, discounted Markov games with bounded rewards. We use two base policies, one for each player, and a model for the Markov game, to estimate the Q-function for the pair of policies. We then use this Q-function for one-step look-ahead, solving the matrix game defined by the Q-function in the current state. Figure 1 displays the pseudo-code for the rollout algorithm for Markov games. In this figure, the function nextState(x, a, b) returns a random state as specified by the transition law of the Markov game, when action a is used by A and action b is used by B in state x. The stochastic algorithm takes two policies $\langle \pi^A, \pi^B \rangle$, an integer N specifying the number of sample paths to be used, a finite horizon H, and the current state x as inputs, and outputs an action $a \in \mathbb{A}$ for player A. The algorithm generates a mixed policy π_{ro} for A. As we will prove shortly, under certain conditions, the policy π_{ro} is better (in security level) than the policy π^A. All the results in this section are stated and proven, without loss of generality, for player A.

In the main result of this section, we bound the state-wise loss in performance due to rollout, and establish overall improvement in the sup norm due to rollout with appropriate choice of π^B, N, and H.

7. The residual of a value function is the sup-norm distance by which the value function shifts when we apply the min-max operator to the value function once.

```
Function: rollout($\pi^A, \pi^B, N, H, x$)
input: policy $\pi^A$ for $A$, policy $\pi^B$ for $B$, number of samples $N$, horizon $H$, state $x$
output: action $a \in \mathbb{A}$

  1. For each pair $\langle a, b \rangle$, $a \in \mathbb{A}, b \in \mathbb{B}$, and $i = 1 \ldots, N$, let
         $\boldsymbol{q}_i(a,b) = R(x,a,b) + \gamma \mathrm{estVal}(\pi^A, \pi^B, H, \mathrm{nextState}(x,a,b))$
  2. Let $\boldsymbol{q}(a,b) = \frac{1}{N} \sum_{i=1} \boldsymbol{q}_i(a,b)$
  3. Return a random action $\boldsymbol{a} \in \mathbb{A}$ according to distribution $\mathrm{Nash}_A(\boldsymbol{q}(\cdot,\cdot))$

Function: estVal($\pi^A, \pi^B, H, x$)
input: policy $\pi^A$ for $A$, policy $\pi^B$ for $B$, horizon $H$, state $x$
output: a sampled estimate of $V_{\pi^A,\pi^B}(x)$

  1. If $H = 0$, return 0
  2. Choose $a$ according to $\mu^{\pi^A}(x)$, and $b$ according to $\mu^{\pi^B}(x)$
  3. Return $R(x,a,b) + \gamma \mathrm{estVal}(\pi^A, \pi^B, H{-}1, \mathrm{nextState}(x,a,b))$
```

Figure 1: The rollout algorithm

4.2 A Policy-improvement Result

In this section we prove that, when called with appropriate parameters, the policy obtained using the rollout pseudo-code shown in Figure 1 is an improvement over the base policy. Note that the rollout algorithm uses sampling to generate a stochastic estimate $\boldsymbol{q}(\cdot,\cdot)$ of $Q_{\pi^A,\pi^B}(x,\cdot,\cdot)$—denote this estimate $\boldsymbol{q}_x(\cdot,\cdot)$. In step 2 of the rollout algorithm, by averaging the N independent estimates $\boldsymbol{q}_x(\cdot,\cdot)$, for each state x, we get a random Q-function $\boldsymbol{Q}_{\mathrm{ro}}$. Let F_{ro} be the distribution for $\boldsymbol{Q}_{\mathrm{ro}}$. Then, the policy π_{ro} generated by the rollout algorithm is $\mathrm{lookahead}_{\mathrm{A}}(F_{\mathrm{ro}})$. Our analysis in this section relies on examining the properties exhibited by $\boldsymbol{Q}_{\mathrm{ro}}$ and its distribution F_{ro}.

Theorem 12 implies that if F_{ro} is a sufficiently accurate approximation of $Q_{\pi^A,\pi^B}(\cdot,\cdot,\cdot)$, for $\pi^B \in \mathrm{brB}(\pi^A)$, then π_{ro} is no worse than π^A in the sup norm. This can be seen by choosing ϵ to be $(1-\gamma)^2 |V^* - V^s_{\pi^A}|_\infty/4$ and δ to be $(1-\gamma)^2 |V^* - V^s_{\pi^A}|_\infty/(8V_{\max})$ in Theorem 12, to get

$$|V^* - V^s_{\pi_{\mathrm{ro}}}|_\infty \quad < \quad \gamma |V^* - V^s_{\pi^A}|_\infty + 2(\epsilon + 2\delta V_{\max})/(1-\gamma) \leq |V^* - V^s_{\pi^A}|_\infty.$$

This inequality is strict, giving a strict contraction, whenever π^A is not already a Nash policy, so that $|V^* - V^s_{\pi^A}|_\infty$ is non-zero.[8]

We now turn to giving sufficient conditions on the sampling horizon H and the number of samples N to achieve the ϵ and δ values just given, so that policy improvement is

8. In addition to this guarantee on the change in the sup-norm, Theorem 12 also provides a bound on the state-wise loss for any choice of ϵ and δ.

guaranteed. Let π^A be a non-Nash policy for A (so that $\left|V^* - V^s_{\pi^A}\right|_\infty > 0$). Let π^B be a corresponding best-response policy. Let π_{ro} be the mixed policy resulting from using rollout(π^A, π^B, N, H, x), at every state x, for some integers N and H. Let ϵ and δ be chosen as above. Now, for any input state x, and for each $\boldsymbol{q}_i(\cdot,\cdot)$ defined in Step 1 of the rollout algorithm (see Figure 1), we have $\left|Q_{\pi^A,\pi^B}(x,\cdot,\cdot) - E\boldsymbol{q}_i(\cdot,\cdot)\right|_\infty \leq \gamma^{H+1}V_{\max} < \gamma^H V_{\max}$. Also, from Proposition 10, it follows that for the stochastic estimate $\boldsymbol{q}_x(\cdot,\cdot)$ defined above, we have $\left|\boldsymbol{q}_x - E\boldsymbol{q}_i\right|_\infty < \epsilon/2$, for any i, with probability at least $1 - e^{-\epsilon^2 N/32V_{\max}^2}$.

We now choose H so that $\left|Q_{\pi^A,\pi^B}(x,\cdot,\cdot) - E\boldsymbol{q}_i(\cdot,\cdot)\right|_\infty < \gamma^H V_{\max} \leq \epsilon/2$, and N so that $\left|\boldsymbol{q}_x - E\boldsymbol{q}_i\right|_\infty < \epsilon/2$, with probability at least $1-\delta$, by ensuring that $e^{-\epsilon^2 N/32V_{\max}^2} < \delta$. With $H > \log(\epsilon/2V_{\max})/(\log\gamma)$ and $N > -32V_{\max}^2(\log\delta)/\epsilon^2$, we then have $|Q_{\pi^A,\pi^B}(x,\cdot,\cdot) - \boldsymbol{q}_x(\cdot,\cdot)|_\infty < \epsilon$ with probability at least $1-\delta$. This holds for every state $x \in \mathbb{X}$. As the random Q-function $\boldsymbol{Q}_{\text{ro}}$ is defined by $\boldsymbol{q}_x$ at each state x, independently, $\boldsymbol{Q}_{\text{ro}}$ satisfies $|Q_{\pi^A,\pi^B}(x,\cdot,\cdot) - \boldsymbol{Q}_{\text{ro}}(x,\cdot,\cdot)|_\infty < \epsilon$ with probability at least $1-\delta$.

Theorem 12 then implies that π_{ro} (i.e., lookahead$_{\text{A}}(F_{\text{ro}})$) is better than π^A in the sup norm, when ϵ, δ, N, and H are chosen to satisfy the following (summarizing the constraints above):

$$\begin{aligned} \epsilon &= (1-\gamma)^2\left|V^* - V^s_{\pi^A}\right|_\infty/4, \\ \delta &= (1-\gamma)^2\left|V^* - V^s_{\pi^A}\right|_\infty/(8V_{\max}) \\ H &> \log(\epsilon/2V_{\max})/(\log\gamma) \\ N &> -32V_{\max}^2(\log\delta)/\epsilon^2. \end{aligned} \tag{4}$$

We have now proven the following theorem.

Theorem 13 *Let π^A be any non-Nash policy for A, and π^B be a corresponding best-response policy. The mixed policy resulting from the rollout algorithm using any values of N and H satisfying the equations in (4) is better than π^A in the sup norm. Such values of the parameters N and H exist and are independent of the size of the state space, $|\mathbb{X}|$.*

In addition to this guarantee of improvement in the sup norm, our main theorem (Theorem 12) guarantees that the rollout policy will have a security level no more than $\frac{2(\epsilon+2\delta V_{\max})}{(1-\gamma)}$ worse that that of the base policy, for any state. This "statewise bound" can be a stronger guarantee than that provided by the sup-norm contraction, and ensures that any part of the state space where the base policy performs well will also be well handled by the rollout policy. Since this term can be made arbitrarily small by choosing large enough sampling width and horizon, we can say that the rollout policy approximately dominates the base policy, at every state, strictly improving at the worst state.

4.3 Discussion

The rollout algorithm presented in this section can be used to improve non-Nash policies. It may appear that, to do so, one requires access to the best-response policy of the opponent, which in general may not be available or effectively computable. However, our theorems do not require an explicit best response policy, but only a distribution approximating the Q-function achieved by it.

If we fix player A's policy, then the Markov game reduces to an MDP, for which the sparse sampling algorithm developed by Kearns et al. (1999) for MDPs can find the best-response Q-values approximately. Our rollout method can use these Q-values to obtain policy improvement, independent of state-space size (noting that the Kearns et al. technique requires exponential effort). The amount of sampling required by the rollout algorithm (number of traces times horizon length) is independent of the state-space size. Hence this algorithm is of particular interest for Markov games with huge state spaces. We believe that, with suitable regularity assumptions, this result extends immediately to continuous state spaces, but we do not explore that claim further here.

It is worth noting that non-Nash policies that are very close to Nash will require large values of N and H to satisfy the equations in (4), and thus will be difficult to improve. Moreover, in practice, it is often difficult to know how far from Nash a given base policy is. Nonetheless, by choosing appropriate values of N and H, we can ensure that we improve the base policy if it is not already within some desired sup-norm distance from Nash-optimal. Our experiments in Section 6 indeed show such improvements in situations where we do not know how far from Nash our base policy is.

5. Sparse Sampling for Markov Games

5.1 The Sparse-sampling Algorithm

Kearns et al. give a sparse-sampling technique for Markov games and prove that the technique computes a near-optimal policy using an amount of sampling independent of the state-space size.[9] But the amount of sampling required is exponential in the desired "accuracy", and hence the policy rollout technique of the previous section is practically more useful. Here, we show that their result is also a direct consequence of our main theorem (Theorem 12), providing a distinct proof. We start by presenting the algorithm carefully, for completeness.

The sparse-sampling algorithm for Markov games, as shown in Figure 2, is straightforward. Again the function $\text{nextState}(x, a, b)$ is used to sample a next state when action a is used by A and action b is used by B in state x. Given the sampling width N (the number of samples at each level), sampling depth H, and the current state x, the algorithm builds a sampling tree (the call tree for estQ^*) to estimate $Q^*(x, \cdot, \cdot)$, the optimal Q-function in the current state, and then solves the resulting matrix game to generate a random action to be taken in state x. Let $\boldsymbol{Q}_{\text{ss}}$ be a random Q-function constructed by combining such independent estimates of $Q^*(x, \cdot, \cdot)$ in all states x (such estimates are obtained by calling the function estQ^* in each state), and let F_{ss} denote its distribution. Then the policy generated by selectAction can be written as $\pi_{\text{ss}} = \text{lookahead}_{\text{A}}(F_{\text{ss}})$. We will show that N and H can be chosen so that the stochastic Q-function $\boldsymbol{Q}_{\text{ss}}$ approximates Q^* with any desired precision. Then, the near-optimality of π_{ss} (the policy generated by the algorithm) follows immediately from Theorem 12.

9. Their algorithm and result is slightly different from ours in that they state and prove their result for general-sum, finite-horizon Markov games, whereas we state and prove our results for zero-sum, discounted Markov games.

```
Function: selectAction$(N, H, x)$
input: sampling width $N$, sampling depth $H$, current state $x$
output: action $a_0$

  1. Return a random action $\boldsymbol{a} \in \mathbb{A}$ according to $\mathrm{Nash}_A(\mathrm{estQ}^*(H, N, x))$


Function: $\mathrm{estQ}^*(H, N, x)$
input: depth $H$, width $N$, state $x$
output: estimated $Q$-function matrix $\hat{\boldsymbol{Q}}(x, \cdot, \cdot)$ for state $x$

  1. If $H = 0$, return zero matrix

  2. For each pair $\langle a, b\rangle$, $a \in \mathbb{A}, b \in \mathbb{B}$, let $\boldsymbol{S}_{a,b}(x)$ be a multiset of $N$ next-state samples
     drawn using nextState$(x, a, b)$

  3. For each pair $\langle a, b\rangle$, $a \in \mathbb{A}, b \in \mathbb{B}$, let
         $\hat{\boldsymbol{Q}}(x, a, b) = R(x, a, b) + \gamma(\sum_{x' \in \boldsymbol{S}_{a,b}(x)} \mathrm{NashVal}(\mathrm{estQ}^*(H-1, N, x')))/N$

  4. return $\hat{\boldsymbol{Q}}(x, \cdot, \cdot)$
```

Figure 2: The sparse-sampling algorithm for Markov games

5.2 Proof of Near-optimality

Now we will prove that algorithm presented in Figure 2 indeed computes a near-Nash policy. While our development is similar to that of Kearns et al. (1999) for MDPs, we deviate from that line of argument by using Theorem 12 (Section 3)—we were unable to use the MDP techniques of Kearns et al. (1999) to prove our result for Markov games here.

Referring to Figure 2, define $\boldsymbol{Q}^h(x, \cdot, \cdot) = \mathrm{estQ}^*(h, N, x)$. Then, for all $h > 0$, $a \in \mathbb{A}$, and $b \in \mathbb{B}$, $\boldsymbol{Q}^0(x, a, b) = 0$ and $\boldsymbol{Q}^h(x, a, b) = R(x, a, b) + \gamma(\sum_{x' \in \boldsymbol{S}_{a,b}(x)} \mathrm{NashVal}(\boldsymbol{Q}^{h-1}(x', \cdot, \cdot)))/N$.

Following Kearns et al. (1999), given some $\lambda > 0$, define $\alpha_0 = V_{\max}$ and α_h recursively as $\alpha_{h+1} = \gamma(\lambda + \alpha_h)$. Then we can bound α_H with

$$\alpha_H = \left(\sum_{i=1}^{H} \gamma^i \lambda\right) + \gamma^H V_{\max} \leq \lambda/(1-\gamma) + \gamma^H V_{\max}. \tag{5}$$

Analogous to Lemma 4 in Kearns et al. (1999), we have the following result. We replicate and adapt their proof, for completeness. To maintain the flow, the proof is postponed to the appendix.

Lemma 14 *With probability at least* $1 - 4(|\mathbb{A}||\mathbb{B}|N+1)^h e^{-\lambda^2 N/(8V_{\max}^2)}$ *we have that* $|Q^*(x, a, b) - \boldsymbol{Q}^h(x, a, b)| \leq \alpha_h$.

We now use Lemma 14 and Theorem 12 to give bounds, independent of the state-space size, on the amount of sampling work needed to give a policy π_{ss} with security level

$V^s_{\pi_{\mathrm{ss}}}(x)$ within ϵ_0 of $V^*(x)$, at each state x for any ϵ_0. We start by choosing values for the parameters ϵ, δ, N, H, and λ of the algorithm, Lemma, and Theorem, in terms of the desired approximation quality, ϵ_0, to meet the following constraints:

$$\begin{aligned}
\epsilon &= (1-\gamma)\epsilon_0/4 && (6)\\
\delta &= (1-\gamma)\epsilon_0/(8V_{\max}) && (7)\\
H &> \log(\epsilon/2V_{\max})/\log\gamma && (8)\\
0<\lambda &< (1-\gamma)\epsilon/2 && (9)\\
N &> \frac{8V_{\max}^2 \log\frac{4(|\mathbb{A}||\mathbb{B}|N+1)^H}{\delta}}{\lambda^2}. && (10)
\end{aligned}$$

Such values exist and are independent of the state-space size.[10] These constraints can be derived by working backwards from the desired policy-quality guarantee through the cited Theorem and Lemma. We now work forwards, using these constraints with the Theorem and Lemma to derive the guarantee. In what follows, we assume that π_{ss} is run using N and H satisfying these constraints.

First, Lemma 14 ensures that we have $|Q^*(x,\cdot,\cdot)-\boldsymbol{Q}_{\mathrm{ss}}(x,\cdot,\cdot)|_\infty \le \alpha_H$ with probability at least $1-4(|\mathbb{A}||\mathbb{B}|N+1)^H e^{-\lambda^2 N/(8V_{\max}^2)}$. Algebraic manipulation of Equation 10 can derive that this probability is greater than $1-\delta$. We can also derive

$$\alpha_H \;\le\; \lambda/(1-\gamma)+\gamma^H V_{\max} \;\le\; \epsilon/2+\gamma^H V_{\max} \;\le\; \epsilon/2+\epsilon/2 \;=\; \epsilon,$$

using first Equation 5 from page 16, then Equation 9, and then Equation 8. We thus have $|Q^*(x,\cdot,\cdot)-\boldsymbol{Q}_{\mathrm{ss}}(x,\cdot,\cdot)|_\infty \le \epsilon$ with probability at least $1-\delta$. We can then apply the sup-norm bound from Theorem 12, choosing π^A to be any Nash equilibrium policy, to derive

$$V^*(x) \;\le\; V^s_{\pi_{\mathrm{ss}}}(x)+2(\epsilon+2\delta V_{\max})/(1-\gamma) \;=\; V^s_{\pi_{\mathrm{ss}}}(x)+\epsilon_0,$$

as desired,where the last equation follows using Equations 6 and 7. We have proven the following theorem.

Theorem 15 *For any $\epsilon_0>0$, the policy π_{ss} generated by* selectAction$(N,H,\cdot)$ *satisfies $|V^*-V^s_{\pi_{\mathrm{ss}}}|_\infty<\epsilon_0$ whenever N and H satisfy Equations 6 through 10 for some values of ϵ, δ, and λ. Moreover, such values of N and H exist and do not depend on the size of the state space, $|\mathbb{X}|$.*

5.3 Discussion

The sparse-sampling algorithm analyzed above computes a near-Nash policy, when sufficient sampling is performed. Unfortunately, even for a discount factor not very close to 1, the sampling required for finding a policy with desirable accuracy could be prohibitively huge. Nevertheless, the amount of sampling needed is independent of the state space size, and hence this technique is of theoretical interest. Moreover, the result presented here just gives a sufficient condition for the policy to be near-Nash. In practice, significantly less sampling, combined with tree-pruning techniques, could result in a useful algorithm (though see Chang (2001) for some network-control applications where this did not prove to be the case). The algorithm is also easily parallelizable.

10. The variable N occurs on both sides of the last inequation, but, for large enough N, the left side grows faster than the right side, guaranteeing that some N satisfying the inequation exists.

6. Empirical Evaluation of Policy Rollout for Markov Games

6.1 Scope of the Simulation Results

The main purpose of the simulation results presented in this section is to demonstrate that the policy-rollout algorithm presented in this paper indeed generates a policy that outperforms the base policy in the sup norm. Even though we provided a theoretical proof of this result, simulation studies are still important for at least the following reasons.

1. We demonstrate that useful improvement can be achieved with a practical amount of sampling, noting that the amount of sampling needed to guarantee improvement is difficult to determine, as we might not know how far we are from the Nash equilibrium, and that that the required sampling, when it can be determined, may be impractical to carry out.

2. We demonstrate that useful improvement can be achieved even without access to a best-response policy for the base policy.

3. We demonstrate that useful improvement can be achieved for other settings, such as Markov games with average-reward criteria or general-sum Markov games.

Clearly, we are interested in policy improvement even when the assumptions of various theorems presented in this paper do not hold true. But as we show using simulation results, performance improvement is still possible in different practical settings. We present two sets of simulation results. First we present the simulation results for a zero-sum, discounted Markov game first introduced by Littman (1994) (the soccer game) and show that using a practically manageable amount of sampling, one can achieve policy improvement in the sup norm in this domain. Next, we apply the policy rollout technique to a bandwidth-market resource-allocation problem presented Savagaonkar (2002) and summarized below. This problem is an interesting test problem in many ways. First, it is a general-sum game, rather than a zero-sum game. Second, the reward criterion is the steady-state reward, rather than discounted reward. We show that, in this case as well, we get policy improvement using practically manageable sampling.

6.2 Soccer

Fig. 3 shows a pictorial view of a soccer-game setting similar[11] to the one presented by Littman (1994). The game is played on a 4×6 grid. The two players, A and B, occupy distinct squares of the grid and can choose one of five actions on each turn: north (N), east (E), south (S), west (W), and stand (X). Once both players have selected their actions, the two moves are executed in random order.

The circle represents the "ball". When the player with the ball steps into the appropriate goal (left for A, right for B), that player scores a point and the board is reset to the configuration shown in Fig. 3, with possession of the ball given to one of the players randomly. When a player attempts to move to the square occupied by the other player, possession of the ball goes to the stationary player and the move does not take place. Goals are worth one point each, and the discount factor is 0.9.

11. The only difference between his game and the game presented here is that his game is played on a grid with five columns, while ours is played on a grid with six columns.

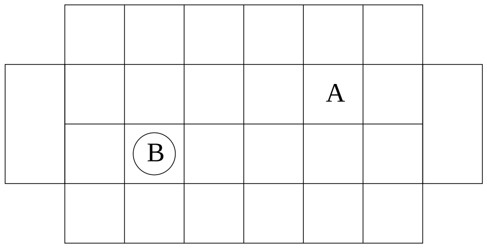


Figure 3: Soccer game similar to the one presented by Littman.

In this setting, we start with a base policy that chooses each of the five possible actions (N, E, S, W, X) with equal probability in each state. We find a best-response policy corresponding to this policy using the value-iteration algorithm for discounted MDPs (once the policy of one player is fixed, the game reduces to an MDP from the other player's perspective). Then, using this policy pair, we invoke the rollout algorithm with N (the number of samples) being 1000 and H (horizon) being 135, starting from every state—this gives us an estimate of the rollout policy (estimating the probability distributions with which actions are chosen for each state). To compute the security level of this policy, we find a best-response policy corresponding to this policy using value iteration—the values computed by value iteration during this process give the security level of the rollout policy.

Fig. 4(a) compares the state-wise loss in the security level for the base policy and the rollout policy[12]. It can be seen that the sup norm of the loss in the security level of the base policy dominates that of the rollout policy by a considerable margin. Fig. 4(b) shows the ratio of the loss in the security level of the rollout policy and that of the base policy, for every state. This ratio is always no larger than 0.81, and generally less than 0.5. Thus, here, the rollout algorithm significantly improves the security level for every state. (Our bound on the state-wise performance only limits how much worse the security level can get, and does not promise an improvement such as we obtain here.)

6.3 Bandwidth Market

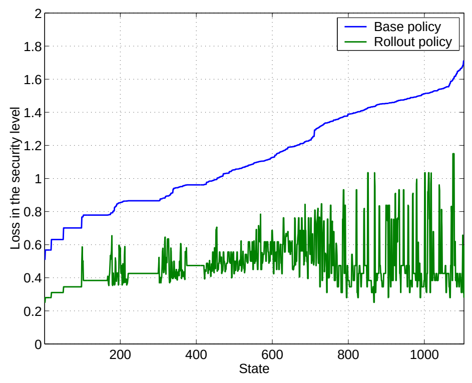
6.3.1 Background

In this section, we focus on a problem of more practical interest—the bandwidth-market problem. This problem was first introduced by Savagaonkar et al. (2002) for bandwidth markets with a single vendor, and later extended to bandwidth markets with multiple vendors (see Savagaonkar, 2002). Below, we briefly describe the "rules" of this bandwidth-market game. The Markov-game formulation of this problem is relatively straight-forward, and is discussed elsewhere (Savagaonkar, 2002).

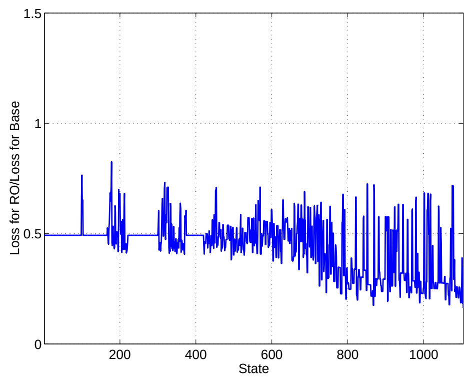
6.3.2 Dynamics of Bandwidth-market Setting

We consider a dynamic market in which two vendors (players) wish to sell bandwidth on two different links, each owned by one vendor. We denote the vendors by A and B. The total amount of bandwidth owned by A is denoted by $\mathcal{B}_A$, and that owned by B is denoted

12. For readability, states are sorted by loss of the base policy in both graphs.



(a)



(b)

Figure 4: (a) Security-level loss of the rollout policy and the base policy. (b) Ratio of the security-level losses of the rollout policy and the base policy.

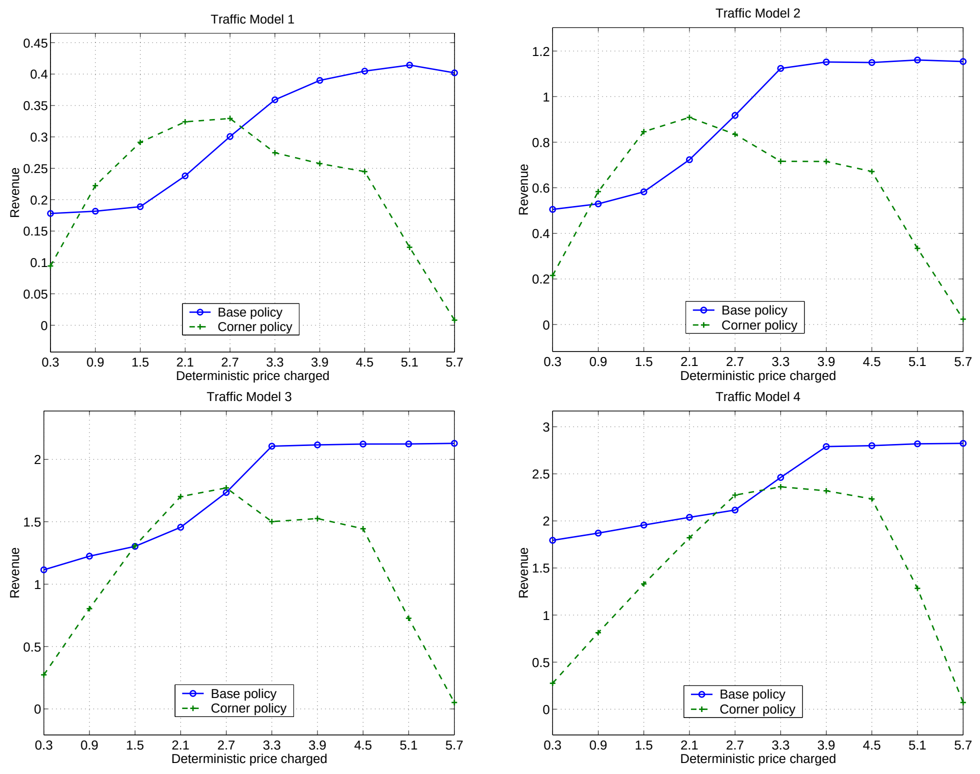


Figure 5: Performance of the base policy when the opponent uses the various corner policies

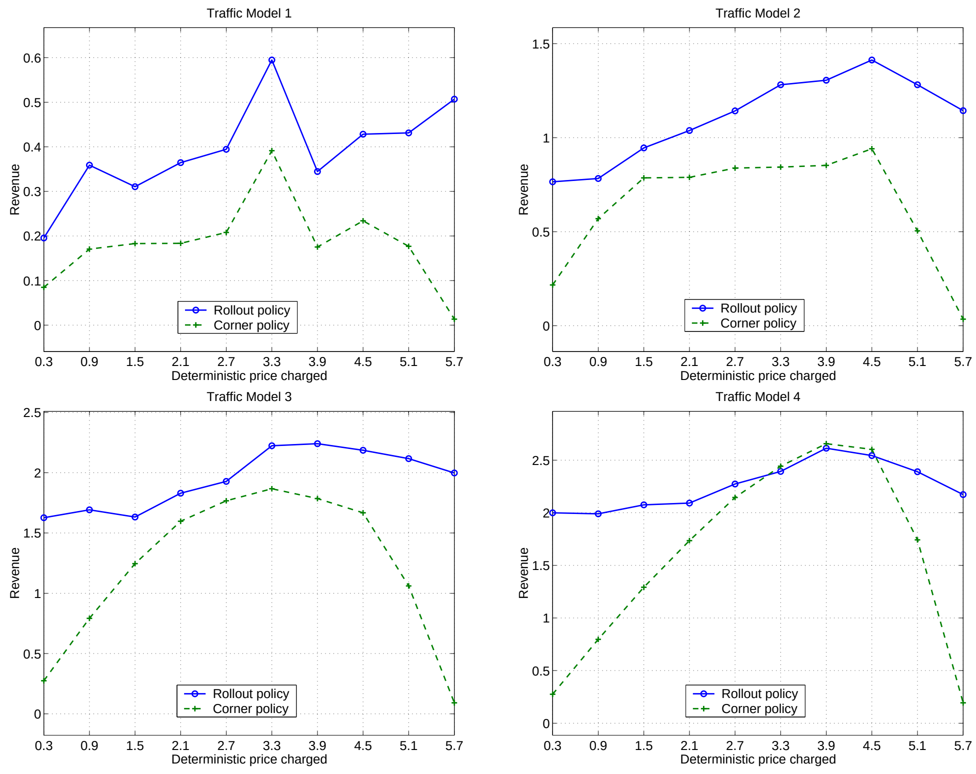


Figure 6: Performance of the rollout policy when the opponent uses the various corner policies

by $\mathcal{B}_B$. We assume that each unit of bandwidth owned by A is equivalent to one unit owned by B. This characterizes our assumption that arriving users do not have any preference of vendor.

We consider a traffic model in which the user arrivals and departures are driven by a discrete-time Markov process $\boldsymbol{S}(\cdot)$ called the *traffic-state process*. We denote the transition-probability matrix of this process by $P(\cdot,\cdot)$, where $P(s,s')$ is the probability that the traffic-state process makes a transition from traffic state s to traffic-state s'. The finite state space, $\mathbb{S}$ of this process is made up of elements called *traffic states*. Each of the users arriving into the system belongs to one of a finitely many classes. We denote the set of all classes by $\mathbb{C}$. We assume that for each state $s \in \mathbb{S}$, the number of calls of class $c \in \mathbb{C}$ arriving in any epoch is a Poisson random variable with mean $\lambda_{s,c}$. We also assume that for each call of class $c \in \mathbb{C}$ arriving in state $s \in \mathbb{S}$, the call-holding time is a geometric random variable with mean $\alpha_{s,c}$. The call-holding time of a call is declared as soon as the call arrives.

We characterize a call i, $i \in \mathbb{Z}_+$, by a triple of random variables, $\langle \boldsymbol{a}_i, \boldsymbol{d}_i, \boldsymbol{c}_i \rangle$, where $\boldsymbol{a}_i$ represents the (integral) time of arrival of call i, $\boldsymbol{d}_i$ represents the (integral) duration of call i, and $\boldsymbol{c}_i$ represents the class of call i.

At each time epoch, bandwidth allocation is performed as follows. Each vendor observes the current *system state*,[13] and declares a price per unit time, per unit of bandwidth it owns. All users belonging to class c are assumed to purchase bandwidth according to a class-specific *demand function* $D_c(\cdot)$ describing the amount of bandwidth desired at each price. We assume that the demand functions are strictly decreasing, strictly concave, and compactly supported (see Savagaonkar, 2002, for related assumptions and justifications). When the vendors declare their prices, the newly arriving users approach the vendor selling the bandwidth at a cheaper price, and declare their demands at this price. If that vendor can provide the requested amount of bandwidth to every user, the users just purchase the bandwidth from this vendor, and do not purchase any bandwidth from the other vendor. However, the cheaper vendor may not have the requested amount of bandwidth available. In that case, the vendor chooses which users to satisfy fully and/or partially, as described below. The unsatisfied users approach the other vendor and request the amount of bandwidth dictated by their demand function at the other vendor's price, less any bandwidth they have already purchased from the cheaper vendor. If we denote by p_l, $l \in \{A, B\}$, the price charged by vendor l at the current decision epoch, then the users request bandwidth according to the following algorithm.

1. Let $i = \arg\min_{l \in \{A,B\}} p_l$ and $j = \arg\max_{l \in \{A,B\}} p_l$.

2. Users in each class c request bandwidth $D_c(p_i)$ from vendor i.

3. Vendor i uses the knowledge of all the requests in the current decision epoch and that of the available bandwidth, and allocates bandwidth b_c^i to users of class c, for each class c (algorithm to be described next). As we will describe shortly, b_c^i is always less than or equal to $D_c(p_i)$.

4. Users in each class c request bandwidth $\max(0, D_c(p_j) - b_c^i)$ from vendor j.

13. The *system state* is defined formally by Savagaonkar (2002) and is part of the Markov-game formulation of this problem.

5. Vendor j uses the knowledge of all the requests in the current decision epoch and that of the available bandwidth, and allocates bandwidth b_c^j to users of class c, for each class c (algorithm to be described next). As we will describe shortly, b_c^j is always less than or equal to $D_c(p_j) - b_c^i$.

When all the users have posted their requests to vendor $l \in \{A, B\}$, the vendor allocates the resource as follows. Let $\mathbb{I}$ denote the set of all the newly arrived users. Let $\tilde{B}_l$ denote the bandwidth available to vendor l. Also, for notational convenience, for each i in $\mathbb{I}$, let R_i^l denote the bandwidth requested by user i to vendor l (algorithm described above). Then the vendor l allocates the bandwidth using the following algorithm.

1. Let $k = \arg\min_{i \in \mathbb{I}} R_i^l$. Break ties using some pre-determined fixed order on classes, selecting within classes arbitrarily.

2. Let $b = \min\{R_k^l, \tilde{B}/|\mathbb{I}|\}$. Allocate bandwidth b to call k.

3. Let $\tilde{B}_l = \tilde{B}_l - b$, and $\mathbb{I} = \mathbb{I} \backslash k$.

4. If $\mathbb{I}$ is not empty, go to step 1.

Each vendor charges each user for the bandwidth it has allocated to that particular user. Bandwidth once sold to a user cannot be reclaimed before the user leaves the system, and the initial allocation to a new user is consumed at every time epoch by that user for the duration of the call. We assume that a user willing to purchase a given bandwidth at a given unit price is also willing to purchase any smaller amount at the same unit price—there is no minimum-bandwidth requirement in our model.

Given the resource-allocation mechanism as described above, the aim of vendor l, $l \in \{A, B\}$, is to set the link prices $p_l(\cdot)$ (a function of discrete time) so that its steady-state revenue is maximized. The revenue obtained by each of the vendors depends, in part, on the actions (prices) chosen by the other vendor. Moreover, it is not always possible to maximize both vendors' revenues simultaneously. Thus, this setting defines a competitive dynamic game.

More precisely, this problem is a general-sum, two-player Markov game, and elsewhere we have developed heuristic pricing schemes and evaluated them on this game (see Savagaonkar, 2002). Below we describe one heuristic pricing scheme–the flat-pricing scheme–briefly, and evaluate the performance of this scheme as well as the pricing scheme obtained by rolling out this scheme.

6.3.3 The Flat-pricing Scheme

Any policy for a vendor specifies a distribution over prices for each state of the game. Now, the assumption of finitely-many classes, when coupled with the assumption of compactly supported demand functions, implies that there exists a price $p_{\max}$, such that every user demands zero resource for any price higher than $p_{\max}$. Thus, a price higher than $p_{\max}$ is not practical for a vendor in any time-epoch. We discretize the range of practical prices (i.e., $[0, p_{\max}]$) into N discrete prices and consider probability distributions over this set of N discrete prices. In this discretized setting, a policy for a vendor specifies a distribution over the finite set of N prices for each possible system state.

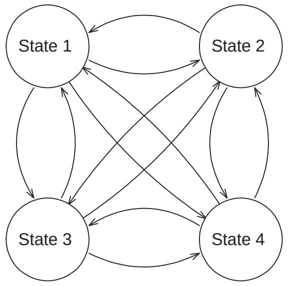


Figure 7: State diagram for the traffic-state process

In the flat-pricing scheme, a vendor fixes a state-independent probability distribution over the above set of prices, and chooses a price according to this same distribution at every epoch. Thus, the flat pricing policy is a parameterized policy in which the probability distribution being used is the parameter—the set of all the flat-pricing policies forms a simplex in an N-dimensional space. The reader is referred to (Savagaonkar, 2002) for a discussion of heuristically finding a locally optimal value of this parameter.[14]

A policy, by which a vendor fixes one of the N discrete prices at every system state, forms a corner of this simplex. We call these policies "corner policies." Below, we measure the performance of any particular pricing scheme based on how it performs when played against each of these N corners, as there is no practical way of finding a best-response policy for an arbitrary pricing policy.

6.3.4 Simulation Results

We assume that the maximum bandwidth available to both the vendors–A and B–is 1. We present the simulation results for a single-class case only, i.e., we assume that all the users arriving into the system have the same demand function–$-\log(p/6)/6$. With this choice of demand function, the value of $p_{\max}$ is 6.

We evaluate the performance of the base policy and the rollout policy for four similar user-arrival models–model 1 through model 4. Each of these four models generates a Markov modulated Poison process (MMPP) with the underlying traffic-state process having four traffic states. Fig. 7 shows the state-transition diagram for the underlying traffic-state process. Table 1 lists the state-transition parameters (the entry in the i^{th} row, j^{th} column indicates the probability of transition from state i to state j) for model 1, and Table 2 lists the traffic parameters describing the traffic statistics for model 1. Models 2 through 4 are derived from model 1 by modifying the ratio of *load factors* in States 1 and 2.

Formally, we define the load factor in traffic state s as the mean call holding time α_s multiplied by the average number of arrivals λ_s in that state (we drop the subscript c representing the class, as we are considering the single-class case). We vary the ratio of the load factors in States 1 and 2 (by varying the load factor in State 2). This change in load factor has an unwanted side effect, described next.

14. Savagaonkar (2002) heuristically uses the algorithm described by Marbach and Tsitsiklis (2001) alternately to find equilibrium price distributions for the two vendors.

Table 1: State-transition probabilities for the traffic-state process

	State 1	State 2	state 3	State 4
State 1	0.95	0.025	0.0125	0.0125
State 2	0.00625	0.9875	0.003125	0.003125
State 3	0.025	0.025	0.55	0.4
State 4	0.025	0.025	0.4	0.55

Table 2: Arrival and holding-time parameters for the traffic-state process

	State 1	State 2	State 3	State 4
λ_s	8	0.25	2	0.25
α_s	0.875	0.875	0.75	0.5

To create other models, we vary the load factor in State 2 in order to explore more and less bursty traffic. To avoid also varying State 2's contribution to the total volume of calls generated, we inversely vary the state holding time in State 2 by varying the self transition probability for State 2.

As mentioned earlier, our price range is $[0, 6]$. For the simulations here, we take N to be ten. We use the heuristic algorithm described by Savagaonkar (2002) to compute a locally optimal flat-pricing policy and a heuristic opponent policy for each of the traffic models, and use those policies as the base policies for the corresponding traffic models. We compute the rollout policy from this policy pair using a rollout horizon of 50, and a sample width of 32. We evaluate both the policies–base and rollout–by measuring their performance against each of the N deterministic flat-pricing opponents (the *corner policies* as described above), as we have no practical means to find a best reponse policy to either of these policies.

Fig. 5 shows the revenue accrued by each player when the base policy plays each corner policy. Fig. 6 shows the revenue accrued by each player when the rollout policy plays the corner policies. Table 3 summarizes these results in the form of the revenue earned by a each policy (base or rollout) when its opponent chooses a corner policy that maximizes its revenue (a rough analog to security levels of the base and rollout policies, for the general-sum context). It can clearly be seen that, in this respect, the rollout policy outperforms the base policy by a significant margin.

Another way of comparing the two policies is to compare the revenue earned by these policies when they are used to play against the opponent's base policy. Table 4 shows this comparison. It can clearly be seen that for all the four traffic models, the rollout policy outperforms the base policy in this metric as well.

Table 3: Performance of the two policies when the opponent chooses the best corner policy

	Base policy	Rollout policy
Traffic Model 1	0.30	0.59
Traffic Model 2	0.72	1.41
Traffic Model 3	1.73	2.22
Traffic Model 4	2.46	2.61

Table 4: Performance of the two policies when the opponent uses the base policy

	Base policy	Rollout policy
Traffic Model 1	0.30	0.47
Traffic Model 2	0.84	1.09
Traffic Model 3	1.71	2.11
Traffic Model 4	2.49	2.57

7. Conclusions

We presented an approximation result for discounted Markov games with bounded rewards. This result establishes a bound on the state-wise loss incurred from using approximate Q-functions for look-ahead. Our result is significantly more general than similar pre-existing results, which cannot be used in the ways we use our result in this paper, and cannot clearly be extended to be so used. Using this result, we discussed two sampling techniques for Markov games. The first technique—policy rollout—is our extension of the policy rollout technique for MDPs to Markov games. We proved that, under appropriate conditions, the policy generated by this technique is closer to the Nash equilibrium than the base policy in the sup norm. We also bound the state-wise loss incurred because of using a (sampled) approximate Q-function during rollout. The second technique is the sparse sampling technique presented by Kearns et al. (2000). We demonstrated the generality of our theorem by providing an alternate proof of Kearns' theorem stating that, with appropriate parameters, this technique produces a policy that is close to the Nash equilibrium with any desired accuracy. For either technique, the amount of sampling required to guarantee the results is independent of the state-space size. Our simulation results indicate that our policy-rollout technique for Markov games indeed provides policy improvement under two interesting settings.

Acknowledgments

This research is supported in part by DARPA/ITO under contract F30602-00-2-0552 and by NSF grants 9977981-IIS, 0093100-IIS, 0098089-ECS, and 0099137-ANI. The views in this document are those of the authors and should not be interpreted as the official policies, expressed or implied, of DARPA, NSF, or the U.S. Government. A preliminary partial version of this paper was presented at the 40th Allerton Conference on Communication, Control, and Computing, October 2002. This research was completed when Uday Savagaonkar was with Purdue University.

References

Noga Alon, Joel H. Spencer, and Paul Erdős. *The Probabilistic Method.* Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wieley and Sons, New York, 1992.

Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory.* Society for Industrial and Applied Mathematics, Philadelphia, 2nd edition, 1995.

R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Volumes 1 and 2.* Athena Scientific, 1995a.

Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1995b.

Dimitri P. Bertsekas and David A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5:89–108, 1999.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic Programming.* Athena Scientific, Belmont, MA 02178, 1996.

Craig Boutilier, Richard Dearden, and Moises Goldszmidt. Stochastic dynamic programming with factored representations. *AIJ*, 121(1-2):49–107, 2000. URL `citeseer.nj.nec.com/boutilier99stochastic.html`.

Craig Boutilier, Raymond Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In *IJCAI*, 2001. URL `citeseer.nj.nec.com/boutilier01symbolic.html`.

Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

Hyeong-Soo Chang. *On-line Sampling-based Control for Network Queuing.* PhD thesis, Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, July 2001.

Edwin K. P. Chong, Robert L. Givan, and Hyeong-Soo Chang. A framework for simulation-based network control via hindsight optimization. *Proceedings of the 39th IEEE Conference on Decision and Control*, 2:1433–1438, 2000.

Alan Fern, SungWook Yoon, and Robert L. Givan. Approximate policy iteration with a policy iteration bias: Learning control knowledge in planning domains. Technical Report TR-ECE-03-11, Purdue University, School of Electrical and Computer Engineering, May 2003. Available from first author's web site.

Robert Givan, Thomas Dean, and Matt Greig. Equivalence notions and model minimization in Markov decision processes. *AIJ*, 147(1-2):163–223, 2003.

R. Howard. *Dynamic Programming and Markov Decision Processes.* MIT Press, 1960.

Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning*, pages 242–250, 1998.

Michael Kearns, Yishay Mansour, and Satinder Singh. Fast planning in stochastic games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 309–316, Morgan Kaufmann, 2000.

Michael J. Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, pages 1324–1231, 1999. URL `citeseer.nj.nec.com/kearns99sparse.html`.

Michail G. Lagoudakis and Ronald Parr. Value function approximation in zero-sum markov games. In *Proceedings of the 18th Conference on Uncertainity in Artificial Intelligence*, Edmonton, Canada, August 2002.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163, 1994.

Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the 18th International Conference on Machine Learning*, pages 322–328, 2001.

Michael L. Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *Proceedings of the 13th International Conference on Machine Learning*, pages 310–318, 1996.

Peter Marbach and John N. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, February 2001.

Hisashi Mine and Shunji Osaki. *Markovian Decision Processes.* American Elsevier Publishing Company, Inc. New York, 1970.

Stephen D. Patek and Dimitri P. Bertsekas. Stochastic shortest path games. *SIAM J. Control Optim.*, 37(3):804–824, 1999.

Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, Inc. New York, 1994.

Uday Savagaonkar. *Network Pricing using Sampling Techniques for Markov Games.* PhD thesis, Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, September 2002.

Uday Savagaonkar, Robert L. Givan, and Edwin K. P. Chong. Dynamic pricing for bandwidth provisioning. In *Proceedings of the 36th Annual Conference on Information Sciences and Systems*, pages 177–182, Princeton, New Jersey, March 2002.

L. Shapley. Multiagent learning using a variable learning rate. *Proceedings of the National Academy of Sciences, USA*, 39:1095–1100, 1953.

Satinder Singh and Richard Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16:227–233, 1994.

G. Tesauro and G. Galperin. On-line policy improvement using monte-carlo search. In *NIPS*, 1996.

S. Yoon, A. Fern, and R. Givan. Inductive policy selection for first-order MDPs. In *UAI*, 2002.

Appendix A. Proofs of Basic Propositions

All our results are proven in the main text, except for these basic proposition, for which proofs (following standard theory) are provided here for completeness. In the following propositions, let $V(\cdot)$ and $V'(\cdot)$ be arbitrary value functions, and $\langle \pi^A, \pi^B \rangle$ be an arbitrary policy pair.

Proposition 2. Suppose $V(x) \leq V'(x)$ for all $x \in \mathbb{X}$. We then have $(T^k V)(x) \leq (T^k V')(x)$ for all $x \in \mathbb{X}$. Also, we have $(T^k_{\pi^A,\pi^B} V)(x) \leq (T^k_{\pi^A,\pi^B} V')(x)$, for all $x \in \mathbb{X}$.

Proof. The proposition follows from the monotonicity of the sum, product, expectation, and max min operators. ■

Recall that we write e for the constant unit value function.

Proposition 3. For any $r \in \mathbb{R}$,

$$\begin{aligned} (T^K(V + re))(x) &= (T^K V)(x) + \gamma^K r, \text{ and} \\ (T^K_{\pi^A,\pi^B}(V + re))(x) &= (T^K_{\pi^A,\pi^B} V)(x) + \gamma^K r. \end{aligned}$$

Proof. The case $K = 0$ is obvious. Then, the proof follows by mathematical induction, noting that the indices achieving the maximum (likewise, minimum) value over an indexed set are not affected by adding a constant term to every member of the set. ■

Proposition 4.

$$\max_{\pi^A} \min_{\pi^B} (T^K_{\pi^A,\pi^B} V)(x) = (T^K V)(x).$$

Proof. The result is obvious for $K = 0$ (both sides are equal to V). Let us assume that the result is true for $K = N - 1$. Then, proving the result for $K = N$ would imply that the result is true for all K.

For $K = N$, the conjecture can be re-written as

$$\max_{\mu_1^A \cdots \mu_{N-1}^A} \max_{\mu_N^A} \min_{\mu_1^B \cdots \mu_{N-1}^B} \min_{\mu_N^B} (T^{N-1}_{\pi^A,\pi^B}(T^N_{\mu_N^A,\mu_N^B} V))(x) = (T^{N-1}(TV))(x). \tag{11}$$

Now, the operator $T^{N-1}_{\pi^A,\pi^B} : \mathbb{V} \to \mathbb{V}$ is completely determined by the maps $\mu_1^A \cdots \mu_{N-1}^A$ and $\mu_1^B \cdots \mu_{N-1}^B$, and thus is independent of μ_N^A, μ_N^A. This implies that the effect of μ_N^A, μ_N^B

on the left-hand side of (11) is reflected only through the term $T^N_{\mu^A_N,\mu^B_N}V$. This observation, when combined with the monotonicity of $T^{N-1}_{\pi^A,\pi^B}$ enables us to write

$$\begin{aligned}
&\max_{\mu^A_1\cdots\mu^A_{N-1}} \max_{\mu^A_N} \min_{\mu^B_1\cdots\mu^B_{N-1}} \min_{\mu^B_N}(T^{N-1}_{\pi^A,\pi^B}(T^N_{\mu^A_N,\mu^B_N}V))(x) \\
&\quad= \max_{\mu^A_1\cdots\mu^A_{N-1}} \max_{\mu^A_N} \min_{\mu^B_1\cdots\mu^B_{N-1}} (T^{N-1}_{\pi^A,\pi^B}\,\mathrm{MIN}_{\mu^B_N}(T^N_{\mu^A_N,\mu^B_N}V))(x) \\
&\quad= \max_{\mu^A_1\cdots\mu^A_{N-1}} \min_{\mu^B_1\cdots\mu^B_{N-1}} (T^{N-1}_{\pi^A,\pi^B}\,\mathrm{MAX}_{\mu^A_N}\,\mathrm{MIN}_{\mu^B_N}(T^N_{\mu^A_N,\mu^B_N}V))(x),
\end{aligned}$$

where the operators MAX and MIN maximize and minimize each component of a value function simultaneously. Note that, as the maps μ^A_N and μ^B_N can assign different probability distributions to each state, independently, these operators are well-defined. Also, because of this same reason, the interchanging of the max and min operators in the last step is justified, as a single map μ^A_N can maximize the objective for all values of $\mu^B_1,\cdots,\mu^B_{N-1}$.

But then, by applying the definition of the operator T, we have that $\mathrm{MAX}_{\mu^A_N}\,\mathrm{MIN}_{\mu^B_N}(T^N_{\mu^A_N,\mu^B_N}V) = TV$. Then, using the notation $TV = V'$, the conjecture can be re-written as,

$$\max_{\mu^A_1\cdots\mu^A_{N-1}} \min_{\mu^B_1\cdots\mu^B_{N-1}} (T^{N-1}_{\pi^A,\pi^B}V')(x) = (T^{N-1}V')(x),$$

which is true by the induction hypothesis. This establishes the result. ■

Proposition 5. The optimal value function V^* is well-defined. Moreover,

$$\lim_{N\to\infty}(T^NV)(x) = V^*(x).$$

Proof. Recall that the security levels for an A-policy π^A and a B-policy π^B are defined as

$$\begin{aligned}
V^s_{\pi^A}(x) &= \min_{\tilde\pi^B} V_{\pi^A,\tilde\pi^B}(x), \text{ and} \\
V^s_{\pi^B}(x) &= \max_{\tilde\pi^A} V_{\tilde\pi^A,\pi^B}(x).
\end{aligned}$$

For every state $x \in \mathbb{X}$, define the optimal security levels

$$\begin{aligned}
\overline{V}^*(x) &= \max_{\pi^A} V^s_{\pi^A}(x), \text{ and} \\
\underline{V}^*(x) &= \min_{\pi^B} V^s_{\pi^B}(x). \qquad (12)
\end{aligned}$$

As the reward function is bounded, these optimal security levels are well-defined. To show that V^* is well-defined, we need to show that for every state $x \in \mathbb{X}$, we have $\overline{V}^*(x) = \underline{V}^*(x)$.

Now, given an A-policy π^A, and any B-policy $\tilde{\pi}^B$, we have

$$\begin{aligned}
V^s_{\pi^A} &\leq \lim_{N\to\infty} E\left\{\sum_{k=0}^{N-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&= E\left\{\sum_{k=0}^{K-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&\quad + \lim_{N\to\infty} E\left\{\sum_{k=K}^{N-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&\leq E\left\{\gamma^K V(\boldsymbol{x}_k) + \sum_{k=0}^{K-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&\quad + \frac{\gamma^K R_{\max}}{1-\gamma} + \gamma^K |V|_\infty \\
&= (T^K_{\pi^A,\tilde{\pi}^B} V)(x) + \frac{\gamma^K R_{\max}}{1-\gamma} + \gamma^K |V|_\infty .
\end{aligned}$$

Taking first $\min_{\tilde{\pi}^B}$ and then taking $\max_{\pi^A}$, and then using Proposition 4, this yields

$$\overline{V}^*(x) \leq (T^K V)(x) + \frac{\gamma^K R_{\max}}{1-\gamma} + \gamma^K |V|_\infty . \tag{13}$$

On the other hand, if $\tilde{\pi}^B$ is a best-response policy for π^A, then we can write,

$$\begin{aligned}
V^s_{\pi^A} &= \lim_{N\to\infty} E\left\{\sum_{k=0}^{N-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&= E\left\{\sum_{k=0}^{K-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&\quad + \lim_{N\to\infty} E\left\{\sum_{k=K}^{N-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\tilde{\pi}^B}(\boldsymbol{x}_k))\right\} \\
&\geq (T^K_{\pi^A,\tilde{\pi}^B} V)(x) - \frac{\gamma^K R_{\max}}{1-\gamma} - \gamma^K |V|_\infty .
\end{aligned}$$

Now, taking $\min_{\tilde{\pi}^B}$ and then taking $\max_{\pi^A}$, and then using Proposition 4, this gives

$$\overline{V}^*(x) \geq (T^K V)(x) + \frac{\gamma^K R_{\max}}{1-\gamma} + \gamma^K |V|_\infty . \tag{14}$$

But this is true for all $\epsilon > 0$, and hence is also true with $\epsilon = 0$. Now (13) and (14) together imply

$$\lim_{K\to\infty} (T^K V)(x) = \overline{V}^*(x). \tag{15}$$

Similarly, it can be shown that

$$\lim_{K\to\infty} (T^K V)(x) = \underline{V}^*(x). \tag{16}$$

Equations (15) and (16) together imply that the two optimal security levels are in fact equal, and hence $V^*(x)$ is well-defined for all $x \in \mathbb{X}$. Also, the convergence of $(T^K V)(x)$ to $V^*(x)$ immediately follows from these equations. ■

Proposition 6.

$$\lim_{N\to\infty} (T^N_{\pi^A,\pi^B} V)(x) = V_{\pi^A,\pi^B}(x).$$

Proof. From the definition of V_{π^A,π^B}, we have

$$\begin{aligned} V_{\pi^A,\pi^B}(x) &= \lim_{N\to\infty} E\left\{ \sum_{k=0}^{N-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\pi^B}(\boldsymbol{x}_k)) \right\} \\ &= E\left\{ \sum_{k=0}^{K-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\pi^B}(\boldsymbol{x}_k)) \right\} \\ &\quad + \lim_{N\to\infty} E\left\{ \sum_{k=K}^{N-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\pi^B}(\boldsymbol{x}_k)) \right\}. \end{aligned}$$

Now as $R \leq R_{\max}$, this implies,

$$\begin{aligned} & V_{\pi^A,\pi^B}(x) - \frac{\gamma^K R_{\max}}{1-\gamma} - \gamma^K |V|_\infty \\ &\leq E\left\{ \gamma^K V(\boldsymbol{x}_k) + \sum_{k=0}^{K-1} \gamma^k R(\boldsymbol{x}_k, \boldsymbol{\mu}_k^{\pi^A}(\boldsymbol{x}_k), \boldsymbol{\mu}_k^{\pi^B}(\boldsymbol{x}_k)) \right\} \\ &= (T^K_{\pi^A,\pi^B} V)(x) \qquad (17) \\ &\leq V_{\pi^A,\pi^B}(x) + \frac{\gamma^K R_{\max}}{1-\gamma} + \gamma^K |V|_\infty, \qquad (18) \end{aligned}$$

which, on taking limit as K goes to infinity yields the result. ■

Proposition 7. Suppose π^A and π^B are stationary. Then the value of the game under this policy pair satisfies $V_{\pi^A,\pi^B} = T_{\pi^A,\pi^B} V_{\pi^A,\pi^B}$.

Proof. From the proof of Proposition 5 (Equation (18)), we have

$$\begin{aligned} & V_{\pi^A,\pi^B}(x) - \frac{\gamma^K R_{\max}}{1-\gamma} - \gamma^K |V|_\infty \\ &\leq (T^K_{\pi^A,\pi^B} V_{\pi^A,\pi^B})(x) \\ &\leq V_{\pi^A,\pi^B}(x) + \frac{\gamma^K R_{\max}}{1-\gamma} + \gamma^K |V|_\infty. \end{aligned}$$

Applying the operator T_{π^A,π^B}, and using Propositions 2 and 3, we get

$$\begin{aligned} & (T_{\pi^A,\pi^B} V_{\pi^A,\pi^B})(x) - \frac{\gamma^{K+1} R_{\max}}{1-\gamma} - \gamma^{K+1} |V|_\infty \\ &\leq (T^{K+1}_{\pi^A,\pi^B} V_{\pi^A,\pi^B})(x) \\ &\leq (T_{\pi^A,\pi^B} V_{\pi^A,\pi^B})(x) + \frac{\gamma^{K+1} R_{\max}}{1-\gamma} + \gamma^{K+1} |V|_\infty. \end{aligned}$$

Which, on taking the limit as K goes to infinity yields the result, using Proposition 7.

∎

Proposition 8. The Nash value satisfies Bellman's equation, $V^* = TV^*$.

Proof. The proof of this proposition is exactly same as that of Proposition 7, except for that we use (13) and (14) instead of (18) and note that $\overline{V}^* = \underline{V}^* = V^*$.

∎

Proposition 9. Suppose V and V' are bounded. For all $k \in \mathbb{N}$, we have

$$\max_{x\in\mathbb{X}} \left|(T^kV)(x) - (T^kV')(x)\right| \le \gamma^k \max_{x\in\mathbb{X}} \left|V(x) - V'(x)\right|.$$

Proof. Let

$$c = \left|V - V'\right|_\infty .$$

Then we have

$$V(x) - c \le V'(x) \le V(x) + c.$$

From Proposition 2 and Proposition 3, we have,

$$(T^kV)(x) - \gamma^k c \le (T^kV')(x) \le (T^kV)(x) + \gamma^k c, \qquad \forall x \in \mathbb{X},$$

from which the result follows. ∎

Proposition 10. Let $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_N$ be i.i.d. random variables satisfying $|\boldsymbol{Y}_i| \le Y_{\max}$ w.p.1 and $E\boldsymbol{Y}_i = \mu$. Then,

$$P\left[\left|\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{Y}_i - \mu\right| \le \lambda\right] \ge 1 - 4e^{-\lambda^2 N/8Y_{\max}^2}.$$

Proof. Our proof follows that by Alon et al. (1992) with small variations. Define a new set of random variables $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ as $\boldsymbol{X}_i = (\boldsymbol{Y}_i - \mu)/Y_{\max}$. Then, $\boldsymbol{X}_i$ are i.i.d., $E\boldsymbol{X}_i = 0$, and $|\boldsymbol{X}_i| \le 1$. Let

$$\boldsymbol{X} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{X}_i.$$

For all $s > 0$, we have

$$\frac{e^s + e^{-s}}{2} \le e^{s^2/2}.$$

This implies,

$$e^s \le 2e^{s^2/2}.$$

Now, as $|\boldsymbol{X}_i| \le 1$ w.p.1, we have for all $s > 0$,

$$E\left[e^{s\boldsymbol{X}_i/N}\right] \le e^{s/N} \le 2e^{s^2/2N^2}.$$

But then,

$$\begin{aligned} E\left[e^{s\boldsymbol{X}}\right] &= \left(E\left[e^{s\boldsymbol{X}_i/N}\right]\right)^N \\ &\le 2e^{s^2/2N}. \end{aligned}$$

Then, using Chernoff bound, we have

$$P\left[\boldsymbol{X} > \lambda\right] = P\left[e^{s\boldsymbol{X}} > e^{s\lambda}\right] \leq 2e^{s^2/2N - s\lambda}.$$

Substituting $s = \lambda N$, we have

$$P\left[\boldsymbol{X} > \lambda\right] \leq 2e^{-\lambda^2 N/2}. \tag{19}$$

Using similar argument, we can also prove that

$$P\left[-\boldsymbol{X} > \lambda\right] \leq 2e^{-\lambda^2 N/2}. \tag{20}$$

Combining (19) and (20), and re-normalizing we get the result. ■

Lemma 14. With probability at least $1 - 4(|\mathbb{A}||\mathbb{B}|N+1)^h e^{-\lambda^2 N/(8V_{\max}^2)}$ we have that $|Q^*(x,a,b) - \boldsymbol{Q}^h(x,a,b)| \leq \alpha_h$.

Proof. The proof is by induction on h, with the base case $h = 0$ being trivially true. Assume for induction that the lemma holds for $h-1$. Recall that $\boldsymbol{f}(x,a,b)$ is a random variable denoting the next state when action pair $\langle a, b\rangle$ is taken in state x. Observe that

$$\begin{aligned}
&|Q^*(x,a,b) - \boldsymbol{Q}^h(x,a,b)| \\
&\quad = \gamma\left(\left|E(V^*(\boldsymbol{f}(x,a,b))) - \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} \mathrm{NashVal}(\boldsymbol{Q}^{h-1}(x',\cdot,\cdot))\right|\right) \\
&\quad \leq \gamma\Bigg(\left|E(V^*(\boldsymbol{f}(x,a,b))) - \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} V^*(x')\right| \\
&\qquad\qquad + \left|\frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} V^*(x') - \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} \mathrm{NashVal}(\boldsymbol{Q}^{h-1}(x',\cdot,\cdot))\right|\Bigg) \\
&\quad \leq \gamma\Bigg(\left|E(V^*(\boldsymbol{f}(x,a,b))) - \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} V^*(x')\right| \\
&\qquad\qquad + \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} \left|\mathrm{NashVal}(Q^*(x',\cdot,\cdot)) - \mathrm{NashVal}(\boldsymbol{Q}^{h-1}(x',\cdot,\cdot))\right|\Bigg) \\
&\quad \leq \gamma\Bigg(\left|E(V^*(\boldsymbol{f}(x,a,b))) - \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} V^*(x')\right| \\
&\qquad\qquad + \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} \left|Q^*(x',\cdot,\cdot) - \boldsymbol{Q}^{h-1}(x',\cdot,\cdot)\right|_\infty\Bigg).
\end{aligned}$$

We now probabilistically bound both terms in the right-hand side of this chain of (in)equations, showing, for $\beta = (|\mathbb{A}||\mathbb{B}|N+1)^{h-1} 4e^{-\lambda^2 N/(8V_{\max}^2)}$, that both

$$P\left(\left|E(V^*(\boldsymbol{f}(x,a,b))) - \frac{1}{N}\sum_{x' \in \boldsymbol{S}_{a,b}(x)} V^*(x')\right| > \lambda\right) < \beta \tag{21}$$

$$P\left(\exists x' \in \boldsymbol{S}_{a,b}(x) \quad \left|Q^*(x',\cdot,\cdot) - \boldsymbol{Q}^{h-1}(x',\cdot,\cdot)\right|_\infty > \alpha_{h-1}\right) < |\mathbb{A}||\mathbb{B}|N\beta. \tag{22}$$

The probability that at least one of the events in Equations 21 and 22 occurs cannot exceed the sum of the individual event probabilites, or $(|\mathbb{A}||\mathbb{B}|N+1)\beta$, which on expanding β is $(|\mathbb{A}||\mathbb{B}|N+1)^h 4e^{-\lambda^2 N/(8V_{\max}^2)}$. But then, the chance that both events do not occur is at least $1-(|\mathbb{A}||\mathbb{B}|N+1)4e^{-\lambda^2 N/(8V_{\max}^2)}$. Then, the above equations guarantee that, with at least this same probability,

$$\begin{aligned}
|Q^*(x,a,b)-\boldsymbol{Q}^h(x,a,b)| &\leq \gamma\Bigg(\Big|E(V^*(\boldsymbol{f}(x,a,b)))-\frac{1}{N}\sum_{x'\in\boldsymbol{S}_{a,b}(x)} V^*(x')\Big| \\
&\qquad + \frac{1}{N}\sum_{x'\in\boldsymbol{S}_{a,b}(x)} \Big|Q^*(x',\cdot,\cdot)-\boldsymbol{Q}^{h-1}(x',\cdot,\cdot)\Big|_\infty\Bigg) \\
&\leq \gamma(\lambda+\alpha_{h-1}) = \alpha_h
\end{aligned}$$

This is our desired result, so it only remains to show Equations 21 and 22.

Equation 21 derives from Proposition 10, which gives

$$\Big|E(V^*(\boldsymbol{f}(x,a,b)))-\frac{1}{N}\sum_{x'\in\boldsymbol{S}_{a,b}(x)} V^*(x')\Big| > \lambda$$

with probability at most $4e^{-\lambda^2 N/(8V_{\max}^2)}$, and thus at most $(|\mathbb{A}||\mathbb{B}|N+1)^{h-1}4e^{-\lambda^2 N/(8V_{\max}^2)}$. Equation 22 follows from the induction hypothesis: we have, for each of $|\mathbb{A}||\mathbb{B}|N$ sampled states $x'\in\boldsymbol{S}_{a,b}(x)$, $|Q^*(x',\cdot,\cdot)-\boldsymbol{Q}^{h-1}(x',\cdot,\cdot)|_\infty > \alpha_{h-1}$ with probability at most $4(|\mathbb{A}||\mathbb{B}|N+1)^{h-1}e^{-\lambda^2 N/(8V_{\max}^2)}$. ■