Sampled Fictitious Play for Approximate Dynamic Programming

Marina Epelman[*], Archis Ghate[†], Robert L. Smith[‡]

January 5, 2011

Abstract

Sampled Fictitious Play (SFP) is a recently proposed iterative learning mechanism for computing Nash equilibria of non-cooperative games. For games of identical interests, every limit point of the sequence of mixed strategies induced by the empirical frequencies of best response actions that players in SFP play is a Nash equilibrium. Because discrete optimization problems can be viewed as games of identical interests wherein Nash equilibria define a type of local optimum, SFP has recently been employed as a heuristic optimization algorithm with promising empirical performance. However there have been no guarantees of convergence to a globally optimal Nash equilibrium established for any of the problem classes considered to date. In this paper, we introduce a variant of SFP and show that it converges almost surely to optimal policies in model-free, finite-horizon stochastic dynamic programs. The key idea is to view the dynamic programming states as players, whose common interest is to maximize the total multi-period expected reward starting in a fixed initial state. We also offer empirical results suggesting that our SFP variant is effective in practice for small to moderate sized model-free problems.

[*]Industrial and Operations Engineering, University of Michigan, Ann Arbor; mepelman@umich.edu
[†]Industrial and Systems Engineering, University of Washington, Seattle; archis@uw.edu
[‡]Industrial and Operations Engineering, University of Michigan, Ann Arbor; rlsmith@umich.edu

1 Introduction

In this paper, we introduce a variant of a game theoretic learning mechanism called Sampled Fictitious Play (SFP) [20] to solve model-free stochastic dynamic programming problems, and investigate its convergence properties and empirical performance. The defining feature of model-free problems is that the state space, immediate rewards resulting from choosing an action in a state, and state transition probabilities are not known explicitly, and hence, system behavior must be "learned" off-line or on-line by repeated computer simulations or system runs. This rules out methods like backward induction, value iteration and mathematical programming. Examples of model-free problems include control of queueing networks with complicated service disciplines whose state transitions are available only through simulation via computer programs [3], control of manufacturing processes where the effect of a decision on the process is calculated by simulating the process [18], and dynamic portfolio optimization or financial derivative pricing problems where the performance of the underlying financial instrument is obtained by simulating complex computer models. Algorithms for model-free problems are termed "simulation based" methods [3, 9, 26, 32] and typically provide an approximate solution. Thus, these simulation based techniques, including our SFP variant, fall within the realm of Approximate Dynamic Programming (ADP) [26].

Stochastic search methods rooted in game theory have recently been applied to large-scale discrete optimization problems, with special focus on cases where the objective function is available only through computationally expensive simulations [2, 10, 14, 15, 16, 20, 21, 22]. Consequently, the hope is to at least find local optima, as stronger forms of optimality are nearly impossible to attain, and very difficult to check. These techniques have been numerically tested with encouraging results on problems in transportation [10, 14, 22], power management in sensor networks [16], network optimization [15], and manufacturing systems [2].

Such heuristic optimization algorithms are applied to problems of the form

$$\max\ u(y^1, y^2, \ldots, y^n)\ \text{ s.t. }\ (y^1, y^2 \ldots, y^n) \in (Y^1 \times Y^2 \times \ldots \times Y^n), \tag{1}$$

where $(Y^1 \times Y^2 \times \ldots \times Y^n)$ denotes the Cartesian product of finite sets Y^1 through Y^n. The main idea is then to animate (1) as a game between n players (corresponding to decision variables $y^1, \ldots, y^n$), who share the *identical interest* of maximizing the objective function $u(\cdot)$. Recall that a Nash equilibrium is a collection of probability distributions over each player's actions with the property that no player can unilaterally improve its utility in expectation by changing its own distribution [13]. Such an equilibrium serves as a type of coordinate-wise local optimum of (1), and hence, the goal is to implement a computationally efficient procedure to find it.

Most of these game theory based techniques for discrete optimization employ variants of Fictitious Play (FP) [6, 29], a well-known iterative learning technique for computing Nash equilibria. At every iteration of FP, each player chooses a strategy that is a best response (with respect to that player's expected utility, which depends on decisions of all players) to the other players' strategies, assuming they will be chosen based on the empirical probability distribution induced by the historical frequency of their best response decisions in all previous iterations. Suitability of the FP approach for optimization stems from its convergence properties. In particular, for a game of identical interests, every limit point of the sequence of mixed strategies induced by the empirical frequencies of best response actions is a Nash equilibrium irrespective of the structure of the objective function [24]. This is termed the "fictitious play property." Another advantage of such methods is that they are easily parallelizable, making them potentially suitable for large-scale optimization problems. However, the best response problem for each player in FP is computationally intractable for realistic problems since an exact computation of the expected utility for a player may require evaluation of the utility function for every possible combination of actions for all players.

As a result, two of the authors and a co-author recently proposed Sampled Fictitious Play (SFP) [20], a modified version of FP where the players choose an action that is a best reply to an independent *sample of actions* of other players drawn according to the empirical probability distribution of actions they used in all previous iterations. SFP offers significant computational advantage over FP, and for games of identical interests, almost surely exhibits the fictitious play property if the sample size is increased at a polynomial rate with iterations [20]. However, efficacy of the original version of SFP for optimization problems has the following significant limitations: (i) the fictitious play property guarantees convergence to only an equilibrium solution rather than an optimal solution, (ii) SFP may converge to a mixed strategy equilibrium, whereas in many applications, and especially in optimization, a pure strategy equilibrium is desirable, (iii) the best response computation becomes increasingly expensive as the sample size grows without bound with iterations, (iv) it is computationally very expensive to force every player in large-scale problems to perform a best reply computation in every iteration, and, finally, (v) problem form (1) excludes optimization problems with constraints across variables. Thus, practical implementations of SFP [2, 10, 21] have attempted ad-hoc variations of the original version. Unfortunately, the original convergence results for SFP in [20] do not hold for these ad-hoc variants. Consequently, the question as to whether one can design an SFP variant that provably finds optimal solutions to an important class of optimization problems by surmounting the above difficulties, has remained open. We answer this question in the affirmative by introducing an SFP variant that solves finite-horizon stochastic dynamic programs.

The key idea is to view the states of the dynamic programming problem as players engaged in a non-cooperative game of *identical interests*, where the objective of each player is to maximize the expected multi-period reward from the *initial state*. The problem structure inherent in dynamic programming, and specifically, the principle of optimality, help our SFP players coordinate their actions and hence solve the problem to optimality. Viewing the states as players also has the important advantage that all combinations of feasible actions of these players are jointly feasible so that the resulting problem is an unconstrained one of the form (1). Importantly, since the objectives of all players are aligned with one another, it suffices for a very small fraction of the players to participate in the game in each iteration, naturally leading to an asynchronous procedure. The procedure to determine which players will participate in an iteration adaptively favors optimal actions (and hence the states they deliver) from the recent past. Specifically, unlike the original SFP in [20], we provide the players with only finite memory. Moreover, we allow players to sample only one action at a time (unlike the original SFP version in [20] which requires an increasing action sample size), and we deliberately add exogenous noise to this selection procedure so that every player in theory gets an opportunity to perform a best response computation infinitely often with probability one even in the asynchronous case. We also remark that if the inherent randomness in state transitions of the system is such that all states will be observed infinitely often with probability one irrespective of the policy implemented, then this exogenous noise is not needed (even in the asynchronous case).

This paper is organized as follows. We develop the necessary notation and formulate our dynamic programming problem in the second section. This is followed by a precise description of our SFP algorithm in the third section. Convergence results and proofs appear in the fourth section. Numerical experiments are presented in the fifth section. Since our SFP variant falls within the realm of simulation-based algorithms for Approximate Dynamic Programming (ADP) [26], a detailed discussion of similarities and differences between our approach and existing simulation-based techniques for ADP is provided in the sixth section, along with other specific conclusions, and future research directions.

2 Problem Formulation

Even though our SFP approach is applicable to any finite-state, finite-action, finite-horizon stochastic dynamic program, it is presented here for the case of model-free problems [3, 26]. In particular, consider the following T period sequential decision problem. The initial state of a system is s_1. In each period $t = 1, 2, \ldots, T$, we observe state s_t of the system and make a decision x_t, which must be chosen from a finite set $X_t(s_t)$ of feasible decisions in state s_t, as determined by a feasibility oracle $\mathcal{F}_t$. This feasibility oracle receives s_t as input and produces a finite set $X_t(s_t)$ as output. Then, another oracle $\mathcal{O}_t$ receives s_t and x_t as input and returns the (stochastic) state of the system s_{t+1} in the next period, and a one period deterministic reward $r_t(s_t, x_t)$. It is common in the literature to consider deterministic one period rewards [7, 33]. The slightly more general case of random one period rewards can also be handled by our algorithm, and the convergence results in Section 4 can be extended to that case without much difficulty. All states of the form s_{T+1} are "terminal" states and there are no feasible actions in these states. We adopt the convention that terminal states have no intrinsic value. Our results generalize in a straightforward manner to problems where nonzero values are assigned to terminal states as for example in our TIC-TAC-TOE example in Section 5. We use S_t to denote the set of feasible states at the beginning of period t, $t = 1, 2, \ldots, T+1$ with $S_1 = \{s_1\}$. Let S denote the finite set of all feasible states of the system, i.e., $S = S_1 \cup S_2 \ldots \cup S_{T+1}$. The sets $S_2, S_3, \ldots, S_{T+1}$, and (hence) the state space S are not known a priori, but must be "discovered" by repeatedly querying oracles $\mathcal{F}_t$ and $\mathcal{O}_t$.

A policy π is a deterministic decision rule that assigns a feasible decision x_t to every state s_t in S. Thus, π is a $|S|$-dimensional vector, and the decision that policy π assigns to state $s_t \in S$ is denoted by $\pi(s_t)$. We use Π to denote the set of all feasible policies of this form. Therefore, for each $s_t \in S$, any policy $\pi \in \Pi$ must have the property that $\pi(s_t) \in X_t(s_t)$. Let $V_\pi(s_1)$ be the value of state s_1 under policy π, i.e., the total T-period expected reward from the initial state if we implement decisions prescribed by policy π in every period. Mathematically,

$$V_\pi(s_1) = r_1(s_1, \pi(s_1)) + E\left(\sum_{i=2}^{T} r_i(s_i, \pi(s_i))\right),$$

where $s_i \in S_i$ for $i = 2, \ldots, T$ and the expectation is with respect to the joint distribution induced by oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_T$. Our goal is to find a policy that solves the problem

$$\pi_* \equiv \underset{\pi \in \Pi}{\operatorname{argmax}}\ V_\pi(s_1). \tag{2}$$

We define $V^*(s_1) \equiv V_{\pi_*}(s_1)$. We assume for simplicity that there is a unique policy π_* optimal to problem (2). The extension to the case of multiple optimal policies is straightforward (see [17]). We next describe our SFP approach in detail.

3 Algorithm Description

Recall that we view the states in S as players engaged in a game of identical interest, namely, that of maximizing the total T-period expected reward from the initial state s_1. Consequently, we henceforth use terms "players" and "states" interchangeably. The central idea in our algorithm is borrowed from SFP: in each iteration, players (i) sample actions according to probability distributions derived from the empirical frequency of their past best responses, (ii) calculate best responses to actions sampled by other players, and (iii) update empirical frequencies of best responses.

3.1 Preliminaries: a synchronous approach

To facilitate discussion, first consider the simpler version of the above model-free problem where the state space and feasible actions are known at the outset, while rewards and state transitions may still be available only through oracles. Suppose the players in this SFP procedure each sample one action in every iteration. These sampled actions then define a policy, and each player can find a best response action simply by comparing an estimate of the value of this policy with the corresponding value estimates of policies that result if the player unilaterally changed its action to each of its other feasible actions. Conceptually, such an SFP procedure would then lead to a synchronous method as outlined in the pseudo-code below:

A Synchronous Version of Sampled Fictitious Play for Approximate Dynamic Programming

1. Action sampling: Each player, independently of the other players, samples one action from its (finite) history of past best responses. (In the first iteration, when the histories are empty, each player samples a feasible action randomly.)

2. Best response: Each player, independently of the other players, estimates (in the manner described in detail in our asynchronous algorithm in Section 3.2) the expected utility (namely the total expected reward from the initial state s_1) corresponding to each one of its feasible actions assuming that the other players have chosen the actions they sampled in Step 1 above, and selects an action that yields the highest reward.

3. History update: Each player updates its own history of best responses by appending it with the best response action chosen in Step 2 above, ensuring through a FIFO procedure that history-length remains less than a preset finite bound.

One can show, using the backward induction technique presented in Section 4, that every entry in every player's history will eventually be optimal with probability one. Thus, we have established that for the class of optimization problems at hand — namely, stochastic finite-horizon dynamic programs — a properly designed variant of SFP finds an optimal solution.

Proposition 3.1. *In the above synchronous Sampled Fictitious Play algorithm for Approximate Dynamic Programming, the probability that there exists an iteration after which all entries in the finite histories of all players are optimal is one.*

Proof. By backward induction as in the proof of Theorem 4.1 in Section 4, and hence omitted. □

However, the above synchronous approach is not viable when the entire state space and feasible actions in each state are not known at the outset. Even if they were available, performing action sampling, value estimation and best response computation for every player in every iteration would be computationally expensive. Thus, the problem setting in Section 2 calls for an asynchronous extension of the above conceptual SFP procedure wherein only some of the players perform action sampling, value estimation and best response calculation in each iteration. However, this introduces new hurdles such as the need to include deliberate exploration to ensure that all players are chosen infinitely often to perform these activities to ensure convergence, and the need to clearly specify a sequence in which players are chosen. Our SFP based asynchronous procedure described below first in text and then as pseudo-code surmounts this difficulty.

3.2 The asynchronous procedure

Each iteration of our asynchronous algorithm consists of two phases: the *choose players* phase and the *best response* phase.

The Choose Players Phase: In the choose players phase, T players, one each from sets $S_1, \ldots, S_T$, are chosen to perform a best response calculation, and are said to be *in play* in that iteration. The precise way in which these players are selected will be explained later. In simple terms, they are selected by drawing a sequence of player-action pairs $\{(s_t, y_t)\}_{t=1}^{T}$ starting at player s_1 and ending at some terminal state, where $y_t \in X_t(s_t)$ is sampled using the history of past plays of player s_t and s_{t+1} is the state obtained by sending state-action pair (s_t, y_t) to oracle $\mathcal{O}_t$. We denote the sequence of player-action pairs drawn this way in iteration k by $\{(s_t^k, y_t^k)\}_{t=1}^{T}$. Observe that corresponding to each player s_t^k in this sequence, there is a realized reward $U_t(s_t^k)$ defined as

$$U_t(s_t^k) = \sum_{i=1}^{t-1} r_i(s_i^k, y_i^k).$$

The choose players phase ends here.

The Best Response Phase: In this phase, each player s_t^k, $t = 1, 2, \ldots, T$ that is selected to be in play in the choose players phase uses each of its feasible actions $x_t \in X_t(s_t^k)$ to generate a "path." A *path* starting at player s_t^k and action $x_t \in X_t(s_t^k)$ is a sequence of player-action pairs $\{(\sigma_{t+j}^k, z_{t+j}^k)\}_{j=1}^{T-t}$, where σ_{t+1}^k is received by sending (s_t^k, x_t) to $\mathcal{O}_t$, player σ_r^k uses its history of past plays to sample action z_r^k for $r = t+1, \ldots, T$, and σ_{r+1}^k is obtained by sending the player-action pair (σ_r^k, z_r^k) to the oracle $\mathcal{O}_r$ for $r = t+1, \ldots, T-1$. We remark that other names for paths in the ADP literature include "episode" and "trajectory" [4, 32]. We denote such a path by $p_k(s_t^k, x_t)$ and define the corresponding reward as

$$R(p_k(s_t^k, x_t)) = \sum_{j=1}^{T-t} r_{t+j}(\sigma_{t+j}^k, z_{t+j}^k). \tag{3}$$

For any player $s_t \in S_t$, and any positive integer m let $I(s_t, m)$ denote the set of iterations in which player s_t is in play up to and including iteration m. We now define the best response problem for player s_t^k that is in play in iteration k. In this iteration k, player s_t^k estimates that if it chooses action $x_t \in X_t(s_t^k)$, its objective, namely, the total T-period expected reward from the initial state s_1, is

$$V^k(s_1; (s_t^k, x_t)) = U_t(s_t^k) + r_t(s_t^k, x_t) + \frac{1}{|I(s_t^k, k)|} \sum_{i \in I(s_t^k, k)} R(p_i(s_t^k, x_t)), \tag{4}$$

where $p_i(s_t^k, x_t)$ is the path drawn by the player-action pair (s_t^k, x_t) in the i^{th} iteration in which player s_t^k was in play. Player s_t^k then solves the following best response problem:

$$\bar{x}^k(s_t^k) \equiv \operatorname*{argmax}_{x_t \in X_t(s_t^k)} \left(V^k(s_1; (s_t^k, x_t)) \right), \tag{5}$$

which is equivalent to the best response problem

$$\bar{x}^k(s_t^k) \equiv \operatorname*{argmax}_{x_t \in X_t(s_t^k)} \left(r_t(s_t^k, x_t) + \frac{1}{|I(s_t^k, k)|} \sum_{i \in I(s_t^k, k)} R(p_i(s_t^k, x_t)) \right) \tag{6}$$

since $U_t(s_t^k)$ does not depend on x_t and therefore there is no need to actually compute $U_t(s_t^k)$ in an algorithmic implementation. Ties in the above problem are broken by choosing an action with the lowest index (note that since there is a finite number of feasible actions in each state, the actions can be indexed). Finally, for brevity, we define

$$W^k(s_t^k, x_t) \equiv \frac{1}{|I(s_t^k, k)|} \sum_{i \in I(s_t^k, k)} R(p_i(s_t^k, x_t)) \tag{7}$$

and rewrite the above best response problem as

$$\bar{x}^k(s_t^k) \equiv \underset{x_t \in X_t(s_t^k)}{\operatorname{argmax}} \Big(r_t(s_t^k, x_t) + W^k(s_t^k, x_t) \Big). \tag{8}$$

Recall that in the Reinforcement Learning [32] parlance, the quantity $r_t(s_t^k, x_t) + W^k(s_t^k, x_t)$ in Equation (8) would be called the "Q-value estimate" (see Equations (13)-(14) below for the definition of "Q-value"). In light of Equation (3) it appears that to use estimation Equation (7), player s_t^k must know $|I(s_t^k, k)|$ and perhaps *every sequence* of player-action pairs $\{(\sigma_{t+j}^i, z_{t+j}^i)\}_{j=1}^{T-t}$ that s_t^k generated in iterations $i \in I(s_t^k, k)$. Although the former is true, the latter is not. To see this, note that $I(s_t^k, k) = I(s_t^k, k-1) \cup k$ and in particular $|I(s_t^k, k)| = 1 + |I(s_t^k, k-1)|$. As a result,

$$\begin{aligned} W^k(s_t^k, x_t) = \frac{1}{|I(s_t^k, k)|} \sum_{i \in I(s_t^k, k)} R(p_i(s_t^k, x_t)) &= \frac{1}{1 + |I(s_t^k, k-1)|} \sum_{i \in \{I(s_t^k, k-1) \cup k\}} R(p_i(s_t^k, x_t)) \\ &= \frac{1}{1 + |I(s_t^k, k-1)|} \left(\sum_{i \in I(s_t^k, k-1)} R(p_i(s_t^k, x_t)) + R(p_k(s_t^k, x_t)) \right) \\ &= \frac{1}{1 + |I(s_t^k, k-1)|} \Big(|I(s_t^k, k-1)| \times W^{k-1}(s_t^k, x_t) + R(p_k(s^k, x_t)) \Big). \end{aligned} \tag{9}$$

Therefore, in order to compute $W^k(s_t^k, x_t)$, player s_t^k needs to know only $|I(s_t^k, k-1)|$, $W^{k-1}(s_t^k, x_t)$ and the reward $R(p_k(s_t^k, x_t))$ associated with the player-action pairs $\{(\sigma_{t+j}^k, z_{t+j}^k)\}_{j=1}^{T-t}$ that it generated in the current iteration k. Based on Equation (9), throughout the algorithm each player s_t stores the number of times it has been in play (denoted by $N(s_t)$) and an estimate $W(s_t, x_t)$ that is updated every iteration s_t is in play as follows:

$$W(s_t, x_t) \leftarrow \frac{(N(s_t) \times W(s_t, x_t) + R(p(s_t, x_t)))}{1 + N(s_t)}. \tag{10}$$

Note that at iteration k, $N(s_t) = |I(s_t, k)|$ is used in the algorithm implementation, whereas our proofs use the notation $I(s_t, k)$ to make the dependence on k explicit. Similarly for $W(s_t, x_t)$ and $W^k(s_t, x_t)$.

Description of History of Past Plays: We define the term "history" here. Let L be a positive integer. For each state $s_t \in S_t$, $t = 1, 2, 3, \ldots, T$, its *history* $H_k(s_t)$ after iteration k is (an ordered list) composed of (at most) L actions in $X_t(s_t)$ that were the solutions to the best response problem for state s_t in the (at most) L most recent iterations in which it was in play. Thus, history of a state is updated each time after it performs a best response computation, and is not updated at any other time. Note that $H_k(s_t)$ is a FIFO list of length at most L; namely, if $|I(s_t, k)| \geq L$, then $|H_k(s_t)| = L$, and if $|I(s_t, k)| < L$, then $|H_k(s_t)| = |I(s_t, k)|$.

Now we are ready to describe exactly how states are chosen in the choose players phase of each iteration to perform a best response computation. This selection is done inductively starting from state s_1 (because that is the only state in S_1). In iteration k, suppose states $s_1, s_2, \ldots, s_t$ have been selected so far. Note that s_t may or may not have a history of past best responses. If its histroy is empty, an action is selected from the set $X_t(s_t)$ of its feasible actions uniformly at random. On the other hand, if the history is not empty, with probability $1 - \alpha_k$ one action is drawn uniformly at random from this history, whereas with probability α_k one action is drawn uniformly at random from $X_t(s_t)$. Suppose this selected action is x_t. Then the pair (s_t, x_t) is sent to the oracle $\mathcal{O}_t$ to receive a state s_{t+1} from set S_{t+1} as output. This is repeated until we reach a terminal state s_{T+1}. The parameter α_k is asymptotically reduced to zero as k increases in a way that ensures that every state is selected infinitely often with probability one. Finally, in the best response phase, the procedure to sample actions (and hence paths) is the same as above except that players do not use parameter α_k while drawing actions when the history is not empty.

Asynchronous Sampled Fictitious Play for Approximate Dynamic Programming

```
1. Initialize: Set $k \leftarrow 1$, choose a decreasing sequence $\{\alpha_k\}_{k=1}^{\infty} \subset [0,1]$, fix some positive integer
   $L$, $N(s_1) \leftarrow 0$, $W(s_1, x_1) \leftarrow 0$ for all $x_1 \in X_1(s_1)$ obtained by sending $s_1$ to oracle $\mathcal{F}_1$, and
   $H_0(s_1) \leftarrow \emptyset$.

2. Choose Players: Starting at player $s_1$, inductively select a sequence of players $s_1 \equiv s_1^k, s_2^k, \ldots, s_T^k$
   said to be in play in iteration $k$, where $s_t^k \in S_t$ for $t = 1, \ldots, T$, as follows:
   for $t = 1, \ldots, T$:
       • If $H_{k-1}(s_t^k) = \emptyset$, then obtain $X_t(s_t^k)$ by sending $s_t^k$ to the feasibility oracle $\mathcal{F}_t$, sample
         an action $y_t^k$ uniformly at random from $X_t(s_t^k)$, set $N(s_t^k) = 0$, and $W(s_t^k, x_t) = 0$ for all
         $x_t \in X_t(s_t^k)$.
       • Else let $u$ be a Uniform $[0,1]$ random variable; if $u \leq 1 - \alpha_k$ sample an action $y_t^k$ uniformly
         at random from $H_{k-1}(s_t^k)$ else sample an action $y_t^k$ uniformly at random from $X_t(s_t^k)$.
       • Send the player-action pair $(s_t^k, y_t^k)$ to oracle $\mathcal{O}_t$ to receive player $s_{t+1}^k$ as output.
   end for.

3. Best Response: For each player $s_t^k$ that is in play:
   (a) for each $x_t \in X_t(s_t^k)$
       draw a path $p_k(s_t^k, x_t) = \{(\sigma_{t+1}^k, z_{t+1}^k), \ldots, (\sigma_T^k, z_T^k)\}$ inductively as follows: send $(s_t^k, x_t)$
       to oracle $\mathcal{O}_t$ to receive $\sigma_{t+1}^k$ as output, and then
           • for $j = 1, \ldots, T - t$:
               – If $H_{k-1}(\sigma_{t+j}^k) = \emptyset$ then sample an action $z_{t+j}^k$ uniformly at random from $X_{t+j}(\sigma_{t+j}^k)$
                 obtained by sending $\sigma_{t+j}^k$ to the feasibility oracle $\mathcal{F}_{t+j}$.
               – Else sample an action $z_{t+j}^k$ uniformly at random from $H_{k-1}(\sigma_{t+j}^k)$.
               – Send the player-action pair $(\sigma_{t+j}^k, z_{t+j}^k)$ to oracle $\mathcal{O}_{t+j}$ to receive player $\sigma_{t+j+1}^k$
                 as output.
           • end for.
       Set $W(s_t^k, x_t) \leftarrow \frac{1}{1+N(s_t^k)} \left( N(s_t^k) \times W(s_t^k, x_t) + R(p_k(s_t^k, x_t)) \right)$.
       end for.
   (b) Find $\bar{x}^k(s_t^k)$ by solving the best response problem (8) where ties are broken via the lowest
       index first rule.
```

(c) Append history $H_{k-1}(s_t^k)$ by $\bar{x}^k(s_t^k)$ in a FIFO manner to keep its length at most L and set $N(s_t^k) \leftarrow N(s_t^k) + 1$.

4. Decrease α_k to α_{k+1}, increase iteration counter to $k+1$ and go to Step 2 (the Choose Players phase).

A few comments on the algorithm are in order.

We will present sufficient conditions on the sequence $\{\alpha_k\}$ to ensure desirable convergence properties in the next section (see Lemma 4.2). It should be noted that deliberate introduction of stochastic noise in choosing players to perform best response computations through parameter α_k is not required in theory if the underlying stochastic behavior of the system itself ensures that every state will be visited infinitely often irrespective of decisions in earlier periods. More generally, the choose players phase and the best response phase are entirely decoupled and the only information the latter needs from the former is a set of players chosen to perform a best response computation. Our convergence results remain valid irrespective of the details of the mechanism used to select this set of players as long as the mechanism ensures that all players are chosen infinitely often with probability one. In the extreme case, every player may be chosen to perform a best response calculation in every iteration resulting in the synchronous procedure outlined in Section 3.1. We reiterate the slight difference between the action sampling mechanisms in the choose players phase and the best response phase: the latter does not employ α_k. This ensures that the estimates $W(s_t, x_t)$ are based on actions that were best responses in earlier iterations, thus avoiding unnecessary stochastic errors in best response calculations. This leads to stronger convergence results and potentially better practical performance.

The idea of using history of finite length in FP-type algorithms has been used with much success in computational game theory in the past (see for example the Adaptive Play (AP) algorithm in [35]). In our context, as mentioned earlier, it is intended to reduce correlation between early and late iterations, progressively producing better value estimates. The stochastic noise and finite history in the sampling process may also be viewed as the players themselves being bounded rational [19].

To assess the computational effort in each iteration, note that since $X_t(s_t)$ is a finite set for all $s_t \in S_t$ and $t = 1, 2, \ldots, T$, there exists a positive integer A such that $|X_t(s_t)| \leq A$ for all t and s_t. Therefore, the number of paths $p_k(s_t^k, x_t)$ that player s_t^k samples in the best response phase is at most A. Each of these paths includes $T - t + 1$ players. The number of players who participate in one iteration is therefore at most $A \sum_{t=1}^{T} (T - t + 1) = AT(T+1)/2$. This observation will be used in comparing our procedure with a benchmark in the numerical results Section 5.

In the parlance of existing simulation-based asynchronous ADP algorithms for model-free finite-horizon problems, Step 2 above can be seen as a mechanism to select states at which policies will be updated. Step 3 (a) can be viewed as an approximate policy evaluation step, whereas Step 3 (b) regarded as a policy improvement mechanism. The probability distribution from which actions will be sampled in the next iteration is updated in Step 3 (c) (indirectly) through changes to the history of plays. In the next two paragraphs, we briefly mention fundamental features of some other simulation-based table look-up methods for the reader to contrast with our SFP procedure. (For a more detailed comparison, and a discussion of convergence results, see Section 6.)

In Monte Carlo methods, the state-action value, that is, Q-value estimate only of each *state-action pair on the selected path* in Step 2 is updated in Step 3 (a) by adding total reward *on that path* following the occurrence of that state-action pair and then smoothing. In comparison, the state-action value estimate of *each state on the selected path* and *each action feasible in that state* is updated in our SFP procedure. Thus the counterpart of Step 3 (a) in typical Monte Carlo methods is simpler than in the pseudo-code above. The decision in each state on the selected path is then set to be optimal with respect to the current Q-value estimates (Steps 3 (b) and 3 (c)). This updated policy is then used in the next iteration to select a new path. See Figure 5.4 in [32].

Temporal Difference methods, unlike Monte Carlo methods, employ bootstrapping to update Q-value estimates of state-action pairs on the selected path. That is, instead of simply adding total reward following the occurrence of a state-action pair on the selected path, Q-value estimates "downstream" on the selected path are used in the update mechanism. See Figure 6.9 in [32]. The well-known Q-learning [34] algorithm also uses bootstrapping and can in fact be viewed as an extension of a particular Temporal Difference method [4]. See Figure 6.12 in [32]. The SFP pseudo-code above, on the other hand, does not bootstrap.

We present convergence results for our asynchronous SFP algorithm in the next section.

4 Convergence Results

We first introduce some notation. We define the value of any state $s_t \in S_t$ for $t = 1, 2, 3, \ldots, T$ under policy $\pi \in \Pi$ as

$$V_\pi(s_t) = r_t(s_t, \pi(s_t)) + E\left(\sum_{i=t+1}^{T} r_i(s_i, \pi(s_i))\right),$$

where the expectation is with respect to the joint distribution induced by oracles $\mathcal{O}_t, \mathcal{O}_{t+1}, \ldots, \mathcal{O}_T$. The corresponding optimal value is defined as

$$V^*(s_t) \equiv \max_{\pi \in \Pi} \; V_\pi(s_t). \tag{11}$$

The above maxima are achieved because the set Π is finite. It is well-known [3, 27] that the optimal values satisfy the following recursive equation

$$V^*(s_t) = \max_{x_t \in X_t(s_t)} \left(r_t(s_t, x_t) + E_{\mathcal{O}_t(s_t, x_t)}[V^*(s_{t+1})]\right), \tag{12}$$

where the subscript on the expectation operator asserts that the expectation is with respect to the probability distribution of state s_{t+1} over S_{t+1} as induced by the stochastic oracle $\mathcal{O}_t$ that receives state-decision pair (s_t, x_t) as input. Note here that $V^*(s_{T+1}) = 0$ for all terminal states s_{T+1} by assumption. The decision prescribed in state s_t by the optimal policy π_* is given by

$$\pi_*(s_t) = \underset{x_t \in X_t(s_t)}{\operatorname{argmax}} \left(\underbrace{r_t(s_t, x_t) + E_{\mathcal{O}_t(s_t, x_t)}[V^*(s_{t+1})]}_{Q(s_t, x_t)}\right), \quad \text{that is,} \tag{13}$$

$$\pi_*(s_t) = \underset{x_t \in X_t(s_t)}{\operatorname{argmax}} \; Q(s_t, x_t). \tag{14}$$

For $t = 1, \ldots, T$ and any $s_t \in S_t$, let $\gamma(s_t)$ be the event that there exists an iteration K_{s_t} such that for all iterations $k \geq K_{s_t}$, each of the L entries in the history of the player corresponding to state s_t is the optimal decision $\pi_*(s_t)$. Similarly, for any $t = 1, \ldots, T$, let $\Omega(t)$ be the event that there exists an iteration K_t such that for all iterations $k \geq K_t$, all L entries in the history of the player corresponding to any state $s \in S_t \cup S_{t+1} \ldots \cup S_T$ equal the optimal action $\pi_*(s)$. Note that $\left(\bigcap_{r=t,\ldots,T}\bigcap_{s_r\in S_r}\gamma(s_r)\right) \Rightarrow \Omega(t)$ since we can choose $K_t = \max_{r=t,\ldots,T}\max_{s_r\in S_r} K_{s_r}$. This implies that $P[\Omega(t)] \geq P\left[\bigcap_{r=t,\ldots,T}\bigcap_{s_r\in S_r}\gamma(s_r)\right]$. Similarly, $\left(\bigcap_{s_{t-1}\in S_{t-1}}\gamma(s_{t-1})\right)\bigcap\Omega(t) \Rightarrow \Omega(t-1)$.

The rest of this section is devoted to the proof of the following Theorem.

Theorem 4.1. $P[\Omega(1)] = 1$, *that is, the probability that there exists an iteration* K_1 *such that for all iterations* $k \geq K_1$ *of asynchronous SFP for ADP, all* L *entries in the histories of players corresponding to all states* $s \in S = S_1 \cup \ldots S_T$ *equal the optimal action* $\pi_*(s)$ *is one.*

We employ an intuitive backward induction proof technique. In particular, we first show that players in S_T are able to compute optimal actions the first time they are in play. Once all players in S_T have been in play at least once (this is ensured by Lemma 4.2), players in S_{T-1} are able to generate increasingly accurate estimates of the total expected reward obtained on playing each of their actions. Consequently, players in S_{T-1} are eventually able to compute their optimal actions after being in play in a sufficiently large number of iterations (this again is ensured by Lemma 4.2). This process is then propagated backwards through periods $T-1, T-2, \ldots, 1$. In particular, after a sufficiently large number of iterations, all best responses are "correct" implying that every entry in the players' history is optimal. These ideas are now formalized.

Observe that, by problem definition, every state $s_{t+1} \in S_{t+1}$ is reachable from the initial state s_1 (otherwise s_{t+1} would not be a member of S_{t+1}). That is, for every $s_{t+1} \in S_{t+1}$, there exists a state-action pair (s_t, x_t) with $s_t \in S_t$ and $x_t \in X_t(s_t)$ such that the probability that oracle $\mathcal{O}_t$ returns state s_{t+1} upon receiving (s_t, x_t) as input is strictly positive. Because we have a finite number of states and actions, there is a uniform (over all periods t, all states s_{t+1}, and all state-action pairs (s_t, x_t)) lower bound $q > 0$ on these strictly positive probabilities.

Lemma 4.2. *If* $\alpha_k \geq (\frac{1}{k})^{1/T}$ *for all iterations* k *then every player is in play infinitely often with probability one.*

Proof. Let $\mathcal{E}^j(s_t)$ be the event that the player corresponding to a specific state $s_t \in S_t$ is in play in iteration j and $t-1$ players in stages $1, 2, \ldots, t-1$ each sampled actions randomly from their sets of feasible actions (as opposed to sampling from their history of best responses) in the choose players phase. Observe that $P[\mathcal{E}^j(s_t)] \geq (q\frac{\alpha_j}{A})^T \geq \left(\frac{q}{A}\right)^T \frac{1}{j}$ and hence $\sum_{j=1}^{\infty} P[\mathcal{E}^j(s_t)] = \infty$. Since events $\mathcal{E}^1(s_t), \mathcal{E}^2(s_t), \ldots$ are independent, the probability that infinitely many of them occur is one by the second Borel-Cantelli lemma [12]. □

Note that a fixed constant strictly positive value of α_k for all k, such as $\alpha_k = 1$ clearly satisfies the sufficient condition in Lemma 4.2 above. However, the importance of decreasing values of α_k is that they employ asymptotically diminishing exploration in the choose players phase, giving more and more weight to best response results of previous iterations along the progress of the algorithm. Potential practical benefits of changing the rate of exploration over iterations instead of using a constant rate have been discussed in the literature, for instance in [32]. An interesting question in our context then is how small a value of α_k can be used still guaranteeing that each player is chosen infinitely often with probability one. Lemma 4.2 provides a simple bound on such a value.

Proof of Theorem 4.1 employs the following inductive statement.

Induction Statement M_t**:** $P[\Omega(t)] = 1$, that is, the probability that there exists an iteration K_t such that for all iterations $k \geq K_t$ of asynchronous SFP for ADP, all L entries in the histories of players corresponding to all states $s \in S_t \cup \ldots S_T$ equal the optimal action $\pi_*(s)$ is one.

Lemma 4.3. *M_T is true, that is $P[\Omega(T)] = 1$.*

Proof. Consider any state $s_T \in S_T$. This player is in play infinitely often with probability one owing to Lemma 4.2. Moreover, every iteration k in which this player is in play, it solves the best reply problem $\bar{x}^k(s_T) = \operatorname*{argmax}_{x_T \in X_T(s_T)} r_T(s_T, x_T)$ since $W^k(s_T, x_T) = 0$ in problem (8) for all $x_T \in X_T(s_T)$. Thus, $\bar{x}^k(s_T) = \pi_*(s_T)$ in every such iteration k owing to the principle of optimality and uniqueness of π_*. Let K_{s_T} be the iteration in which this player is in play for the Lth time (such an iteration exists with probability one, again owing to Lemma 4.2). Then for all iterations $k \geq K_{s_T}$, each of the L entries in this player's history is $\pi_*(s_T)$. Since $s_T \in S_T$ was arbitrary, $P[\gamma(s_T)] = 1$ for every $s_T \in S_T$. Then it is easy to see that $P\Big[\bigcap_{s_T \in S_T} \gamma(s_T)\Big] = 1$. This implies that $P[\Omega(T)] = 1$. $\square$

Now we prove an intermediate lemma.

Lemma 4.4. *M_t implies that for every player-action pair (s_{t-1}, x_{t-1}) with $s_{t-1} \in S_{t-1}$ and $x_{t-1} \in X_{t-1}(s_{t-1})$, the estimate $W^k(s_{t-1}, x_{t-1})$ converges to $E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)$ as $k \to \infty$ with probability one. As a result, $r_{t-1}(s_{t-1}, x_{t-1}) + W^k(s_{t-1}, x_{t-1})$ converges to $r_{t-1}(s_{t-1}, x_{t-1}) + E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)$ as $k \to \infty$ with probability one.*

Proof. Recall that $I(s_{t-1}, k)$ is the set of iterations up to and including iteration k in which the player corresponding to state s_{t-1} is in play. Let $J(s_{t-1}, m, n)$ be the set of iterations between iterations m and n (not including m but including n) in which the player corresponding to state s_{t-1} is in play. Let K_t be as defined in the induction statement above and $k \geq K_t$ be any iteration large enough such that $|I(s_{t-1}, k)| > 0$ (such an iteration exists with probability one by Lemma 4.2). We have

$$|W^k(s_{t-1}, x_{t-1}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)|$$

$$= \left| \frac{1}{|I(s_{t-1}, k)|} \sum_{i \in I(s_{t-1},k)} \sum_{j=0}^{T-t} r_{t+j}(\sigma^i_{t+j}, x^i_{t+j}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right|.$$

Using $I \equiv I(s_{t-1}, K_t)$ and $J \equiv J(s_{t-1}, K_t, k)$ for brevity, the above right hand side becomes

$$= \left| \frac{1}{|I| + |J|} \sum_{i \in \{I \cup J\}} \sum_{j=0}^{T-t} r_{t+j}(\sigma^i_{t+j}, x^i_{t+j}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right|,$$

which in turn is bounded above by

$$\left| \frac{1}{|I| + |J|} \sum_{i \in I} \sum_{j=0}^{T-t} r_{t+j}(\sigma^i_{t+j}, x^i_{t+j}) \right| + \left| \frac{1}{|I| + |J|} \sum_{i \in J} \sum_{j=0}^{T-t} r_{t+j}(\sigma^i_{t+j}, x^i_{t+j}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right|.$$

Since the state and decision spaces are finite, the absolute values of one period deterministic rewards are uniformly bounded above, say by a positive number R. As a result, the above right hand side is at most

$$\left| \frac{|I| R (1 + T - t)}{|I| + |J|} \right| + \left| \frac{1}{|I| + |J|} \sum_{i \in J} \sum_{j=0}^{T-t} r_{t+j}(\sigma^i_{t+j}, x^i_{t+j}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right|$$

$$= \left| \frac{|I| R (1 + T - t)}{|I| + |J|} \right| + \frac{|J|}{|I| + |J|} \left| \frac{1}{|J|} \sum_{i \in J} \sum_{j=0}^{T-t} r_{t+j}(\sigma^i_{t+j}, x^i_{t+j}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right|.$$

The first term above converges to zero as $k \to \infty$ with probability one since $J(s_{t-1}, K_t, k) \to \infty$ as $k \to \infty$ with probability one owing to Lemma 4.2. Similarly, $\frac{|J|}{|I|+|J|} \to 1$ as $k \to \infty$ with probability one. Thus we only focus on the remaining term

$$\left| \frac{1}{|J|} \sum_{i \in J} \sum_{j=0}^{T-t} r_{t+j}(\sigma_{t+j}^i, x_{t+j}^i) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right|.$$

Note that M_t implies that for all iterations $i \in J(s_{t-1}, K_t, k)$, $x_{t+j}^i = \pi_*(\sigma_{t+j}^i)$ for $j = 0, \ldots, T-t$. Moreover, when state-decision pair (s_{t-1}, x_{t-1}) is sent to the oracle $\mathcal{O}_{t-1}$, the sums

$$\sum_{j=0}^{T-t} r_{t+j}(\sigma_{t+j}^i, \pi_*(\sigma_{t+j}^i))$$

are independent and identically distributed for iterations $i \in J(s_{t-1}, K_t, k)$. Therefore,

$$\frac{1}{|J|} \sum_{i \in J} \sum_{j=0}^{T-t} r_{t+j}(\sigma_{t+j}^i, x_{t+j}^i) \to E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \text{ as } k \to \infty$$

with probability one by the strong law of large numbers and the definition of $V^*(s_t)$ for any $s_t \in S_t$. Thus the above term of interest converges to zero with probability one. Thus $|W^k(s_{t-1}, x_{t-1}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)|$ is bounded below by zero and bounded above by a term that converges to zero as $k \to \infty$ with probability one. Therefore, $W^k(s_{t-1}, x_{t-1})$ converges to $E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)$ as $k \to \infty$ with probability one. This completes the proof. □

The above intermediate lemma helps us restore the inductive hypothesis as follows.

Lemma 4.5. *M_t implies M_{t-1}.*

Proof. Consider $\delta(s_{t-1})$ defined as follows:

$$\delta(s_{t-1}) = \min_{\pi_*(s_{t-1}) \neq x_{t-1} \in X_{t-1}(s_{t-1})} \left(V^*(s_{t-1}) - r_{t-1}(s_{t-1}, x_{t-1}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) \right),$$

that is, $\delta(s_{t-1})$ is the difference between the optimal value of state s_{t-1} and the value of the next-best feasible action in s_{t-1}. The above minimum is well-defined since $X_{t-1}(s_{t-1})$ is a finite set. The definition of $V^*(s_{t-1})$ implies that $\delta(s_{t-1}) \geq 0$ whereas uniqueness of the optimal policy implies that $\delta(s_{t-1}) > 0$. The best reply problem (8) for the player corresponding to state s_{t-1} implies that if $|r_{t-1}(s_{t-1}, x_{t-1}) + E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t) - r_{t-1}(s_{t-1}, x_{t-1}) - W^k(s_{t-1}, x_{t-1})| < \delta(s_{t-1})/2$ for all $x_{t-1} \in X_{t-1}(s_{t-1})$, then $\bar{x}^k(s_{t-1}) = \pi_*(s_{t-1})$. For any $\epsilon > 0$, let $A_\epsilon(s_{t-1}, x_{t-1})$ be the event that there exists an integer $N_\epsilon(s_{t-1}, x_{t-1})$ such that for all iterations $k \geq N_\epsilon(s_{t-1}, x_{t-1})$,

$$|r_{t-1}(s_{t-1}, x_{t-1}) + W^k(s_{t-1}, x_{t-1}) - r_{t-1}(s_{t-1}, x_{t-1}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)| < \epsilon.$$

Note that the almost sure convergence

$$r_{t-1}(s_{t-1}, x_{t-1}) + W^k(s_{t-1}, x_{t-1}) \to r_{t-1}(s_{t-1}, x_{t-1}) + E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})} V^*(s_t)$$

is equivalent (see [30]) to $P[A_\epsilon(s_{t-1}, x_{t-1})] = 1$. Therefore,

$$P\left[\bigcap_{x_{t-1} \in X_{t-1}(s_{t-1})} A_{\frac{\delta(s_{t-1})}{2}}(s_{t-1}, x_{t-1}) \right] = 1.$$

More specifically, choosing $N_{\frac{\delta(s_{t-1})}{2}}(s_{t-1}) = \max_{x_{t-1}\in X_{t-1}(s_{t-1})} N_{\frac{\delta(s_{t-1})}{2}}(s_{t-1}, x_{t-1})$, the probability that

$$|r_{t-1}(s_{t-1}, x_{t-1}) + W^k(s_{t-1}, x_{t-1}) - r_{t-1}(s_{t-1}, x_{t-1}) - E_{\mathcal{O}_{t-1}(s_{t-1},x_{t-1})}V^*(s_t)| < \frac{\delta_{s_{t-1}}}{2}$$

for all $x_{t-1} \in X_{t-1}(s_{t-1})$ and for all iterations $k \geq N_{\frac{\delta(s_{t-1})}{2}}(s_{t-1})$ is one. As a result, since the player corresponding to state s_{t-1} is in play infinitely often with probability one, the probability that there exists an iteration $K'_{s_{t-1}}$ such that $\bar{x}^k(s_{t-1}) = \pi_*(s_{t-1})$ for all iterations $k \geq K'_{s_{t-1}}$ is one. This implies that $P[\gamma(s_{t-1})] = 1$. Then it is easy to see that $P\left[\bigcap_{s_{t-1}\in S_{t-1}} \gamma(s_{t-1})\right] = 1$. Together with M_t, i.e., $P[\Omega(t)] = 1$, this implies that $P[\Omega(t-1)] = 1$. That is, M_{t-1} holds. $\square$

By the principle of induction, Lemmas 4.3 and 4.5 imply that M_1 is true, that is, $P[\Omega(1)] = 1$. This concludes the proof of Theorem 4.1.

Since $M_T, \ldots, M_t, \ldots, M_1$ are true, Lemma 4.4 implies that for any $t = 1, \ldots, T-1$, and any state-decision pairs (s_t, x_t) with $s_t \in S_t$ and $x_t \in X_t(s_t)$, $W^k(s_t, x_t)$ converges to $E_{\mathcal{O}_t(s_t,x_t)}V^*(s_{t+1})$ as $k \to \infty$ with probability one. In addition, Theorem 4.1 above states that with probability one, there exists an iteration K_1 such that for all iterations $k \geq K_1$, every entry in the history of the player corresponding to state s_1, denoted by $h_k(s_1)$, is $\pi_*(s_1)$. We thus have the following "value convergence" result.

Corollary 4.6. *$r_1(s_1, h_k(s_1)) + W^k(s_1, h_k(s_1))$ converges with probability one to $V^*(s_1)$ as $k \to \infty$.*

Let $E^k(s_t)$ denote the event that state $s_t \in S_t$ is in play in iteration k. Given event $E^k(s_t)$, let $y^k(s_t)$ denote the feasible action sampled by state s_t in the choose players phase of the SFP algorithm of Section 3.2. Similarly, let $D^k(s_t)$ denote the event that state s_t gets the opportunity to sample an action in the best response phase in iteration k, and given $D^k(s_t)$, let $z^k(s_t)$ denote an action sampled by s_t. Then Theorem 4.1 and the ideas discussed in its proof imply the following "solution convergence" result.

Corollary 4.7. *For any $0 < \epsilon < 1$, there exists an iteration K_ϵ such that $P[y^k(s_t) = \pi^*(s_t)|E^k(s_t)] > 1 - \epsilon$ for all $k \geq K_\epsilon$. That is, sufficiently far along the progress of the algorithm, the actions sampled by players in the choose players phase are optimal with arbitrarily high probability. Similarly, $P[z^k(s_t) = \pi^*(s_t)|D^k(s_t)] = 1$ for sufficiently large k. That is, sufficiently far along the progress of the algorithm, the actions sampled by players in the best response phase are optimal.*

In summary, the version of SFP designed here uses action samples of size one, finite history of past best responses, and allows players to make sampling mistakes. It is much simpler to implement in practice and computationally more efficient than the original approach in [20], and yet exhibits stronger convergence behavior. In the next section, we present numerical results that suggest our asynchronous SFP procedure is effective in practice.

5 Numerical Results

We apply SFP to two stochastic inventory control examples from [7] and to the game of TIC-TAC-TOE against nature. The state spaces in the inventory control examples are small, whereas in TIC-TAC-TOE the state space is moderate.

5.1 Stochastic Inventory Control Example 1

This example is taken from [7]. Our main reason for choosing this example is the same as given in [7], namely, that it can be solved either by using inventory theory or by stochastic dynamic programming backward recursion as described in [7], and hence it is easy to test the performance of SFP. Moreover, it provides us the opportunity to compare the performance of SFP with that of a recent simulation-based method for stochastic dynamic programming.

Consider the following T period inventory control problem. At the beginning of period $t = 1, 2, \dots, T$, we observe inventory level s_t of a single product and decide whether to order an amount q of the product or not. The ordered amount arrives immediately. The inventory levels are bounded above by a positive integer M. Note that an order can be placed only if $s_t + q \leq M$. A random demand D_t is realized after receiving the placed order. The distribution of this demand is not explicitly specified to the decision maker, however, it is known that demand is an integer from the set $\{0, 1, \dots, D\}$, where D is a positive integer. The inventory s_{t+1} at the beginning of the next period is given by $s_{t+1} = \max\{s_t + x_t - D_t, 0\}$, that is, backlogging is not allowed and unsatisfied demand is lost. We incur three types of cost. There is a fixed cost of K every time we place an order, that is, every period t in which $x_t = q$. Every unit of inventory left over at the end of period t is charged an inventory holding cost of h. There is also a penalty cost of p per unit of lost demand. The initial inventory at the beginning of the first period is s_1. The objective is to decide the order quantity for every possible inventory level in each period so as to minimize the expected total cost over the entire horizon of T periods. Observe that this problem is slightly more general than the generic formulation described in Section 2 in that one period cost for a given state-action pair is random. In particular, the one period cost for ordering a feasible amount x_t when the inventory is s_t is given by

$$K \times I(x_t > 0) + h \times (s_t + x_t - D_t)^+ + p \times (s_t + x_t - D_t)^-,$$

where $I(x_t > 0)$ is the indicator function which equals one if $x_t > 0$ and zero otherwise, $(s_t + x_t - D_t)^+ = \max\{0, (s_t + x_t - D_t)\}$, and $(s_t + x_t - D_t)^- = \max\{0, -(s_t + x_t - D_t)\}$. We solve this problem using the following data from [7]: $T = 3$, $q = 10$, $M = 20$, $h = 1$, and the demand is a uniformly chosen integer from $[0, 9]$. SFP does not use this information about the demand distribution, however this information is needed in our numerical experiments to implement the oracles $\mathcal{O}_t$. We use two value of $K = 0, 5$ and two values of $p = 1, 10$. The initial inventory is $s_1 = 5$. We wish to estimate $V^*(s_1)$.

It is important to reiterate that even though the demand distribution is known to be uniform, that information is used only by the backward recursion algorithm we employed to calculate optimal values to be compared with the results of our SFP procedure. The SFP procedure does not use this explicit demand distribution information but rather "learns" the distribution through repeated samples generated from it. As a result, the challenge in this model-free version of the stochastic inventory control problem with small action and state spaces is in accurately estimating the expected rewards and then optimizing the ordering policy based on these estimates.

The first column in Table 1 below lists the values of K and p used. The second column lists the corresponding optimal value $V^*(s_1 = 5)$. The third column reports the average of the optimal values estimated by our algorithm after 50 iterations over 30 independent runs when the players remember only their most recent best response, i.e., $L = 1$. The fourth column lists this average when the players remember their five most recent best responses, i.e., $L = 5$. Parameter α_k was taken to be $(1/k)^{1/T}$ for all iterations k in all runs as it satisfies the rate calculated in Lemma 4.2. In order to estimate the optimal value of state s_t, the algorithm in [7], which the authors term AMS for Adaptive Multistage Sampling, recursively samples N states from S_{t+1} for each action in s_t and proposes three different estimators: the first estimator uses a weighted average of Q-values, the second estimator uses the maximum Q-value whereas the third one employs the Q-value of the most frequently sampled action. For comparison, we have copied from [7] estimates of the optimal value of state $s_1 = 5$ using these three estimators with sample sizes $N = 4$ and $N = 32$ (these were the smallest and the largest sample sizes used, respectively) in the last three columns of Table 1. The evolution of our estimate of $V^*(s_1 = 5)$ is plotted versus 50 iterations in Figure 1 and Figure 2.

The runtimes of SFP and the AMS algorithm of [7] can be assessed by looking at the number of states sampled by these methods, since sampling is the bottleneck step of both algorithms. This method of analysis provides a reasonable comparison of the algorithms, including the rate of growth of runtime as number of iterations or problem size increases, without focusing on the specifics of implementation or computers used. Since the optimal value of a state in AMS is recursively estimated by sampling N states from the next stage for each action in the state at hand, the number of states sampled in a problem with A actions, horizon T and sample size N is $(AN)^T$ (page 129 of [7]). Since $A = 2$ and $T = 3$ in this inventory control example, the number of states sampled by AMS is 512 with $N = 4$ and 262,144 with $N = 32$. Recalling the discussion of the asynchronous algorithm in Section 3.2, the number of states sampled by SFP in 50 iterations is $50 \times AT(T + 1)/2 = 50 \times 2 \times 12/2 = 600$, and hence its computational effort for this problem is comparable to the faster version ($N = 4$) of AMS.

Comparing the optimal value estimates obtained by SFP to those obtained by AMS using $N = 4$, which requires similar computational effort, we observe (Table 1) that, for the cases $(K, p) = (0, 1)$, $(0, 10)$, $(5, 10)$, SFP estimates are significantly closer to the optimal value than those obtained by AMS with either of the three estimators. Comparing SFP estimates for these three cases to AMS with $N = 32$, we see that SFP estimates are more accurate than those obtained with the first estimator and only slightly worse than ones obtained with the other two estimators, even though SFP requires only 0.23% of the samples that AMS with $N = 32$ does. When $(K, p) = (5, 1)$, the SFP estimate outperforms the first estimator with $N = 4$, but is worse in other cases. Finally, the tables indicate that SFP performance is not very different for $L = 1$ and $L = 5$, although $L = 1$ seems to work slightly better for this problem.

Table 1: Estimates $V(s_1 = 5)$ of $V^*(s_1 = 5)$ for stochastic inventory control Example 1 averaged over 30 independent runs of 50 iterations of SFP with $L = 1$ and $L = 5$. The three estimates reported in [7] each with sample sizes $N = 4$ and $N = 32$ are listed in the last three columns for comparison.

		SFP estimates		AMS estimates from [7]		
(K,p)	$V^*(s_1 = 5)$	$L = 1$	$L = 5$	Estimate 1 N=4, N=32	Estimate 2 N=4, N=32	Estimate 3 N=4, N=32
(0,1)	10.440	10.8856	11.4699	15.03, 11.23	9.13, 10.45	9.56, 10.49
(0,10)	24.745	24.9614	25.7516	30.45, 26.12	19.98, 24.73	20.48, 24.74
(5,1)	10.490	12.4176	12.3314	18.45, 11.47	10.23, 10.46	10.41, 10.46
(5,10)	31.635	31.183	32.6679	37.52, 33.11	26.42, 31.62	26.92, 31.64

Like any other finite-horizon dynamic programming algorithm, the computational effort in our procedure depends on the length of problem horizon. This is illustrated for the inventory control example at hand in Table 2. The table lists average of our optimal value estimate over 30 independent runs of 50 iterations each as in Table 1 for $L = 1$ and various values of T. Optimal values are also listed for comparison. The percentage relative error in our optimal value estimates

Table 2: Sensitivity of the accuracy of optimal value estimates to horizon T in stochastic inventory control Example 1. The first column lists various horizon lengths. The other columns present the optimal values $V^*(s_1 = 5)$, and optimal value estimates $V(s_1 = 5)$ averaged over 30 independent trials of 50 iterations of SFP with $L = 1$ (separated by a comma) for different combinations of (K, p). The percentage relative error between $V^*(s_1 = 5)$ and $V(s_1 = 5)$ is plotted in Figure 3. Note that the first row of results is copied from Table 1.

T	$(K = 0, p = 1)$	$(K = 0, p = 10)$	$(K = 5, p = 1)$	$(K = 5, p = 10)$
3	10.440, 10.8856	24.745, 24.9614	10.490, 12.4176	31.635, 31.183
4	14.4752, 14.3797	31.8861, 31.6529	14.9452, 17.6562	40.9761, 40.9536
5	18.0422, 18.5967	39.0273, 39.0144	19.4368, 23.6745	50.3198, 50.7654
6	22.5475, 23.0523	46.1685, 47.4667	23.9345, 29.498	59.6634, 59.5954
7	26.587, 27.8457	53.3097, 54.4915	28.4351, 34.5876	69.0071, 70.549
8	30.6267, 32.6117	60.4509, 62.0641	32.9351, 39.6497	78.3507, 80.1529
9	34.6663, 36.9863	67.5921, 70.732	37.4351, 43.9118	87.6944, 89.0752
10	38.706, 42.1640	74.7333, 79.6176	41.9351, 50.7621	97.038, 101.505

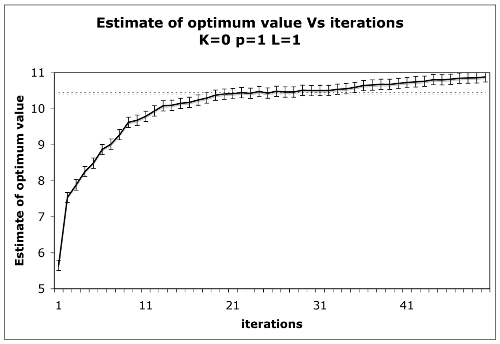
after 50 iterations is plotted versus horizon length in Figure 3. We remark that our algorithm was able to reach closer to the optimal values in all cases after running more iterations; however, this is not apparent in Table 2 since we deliberately fixed the iterations to 50 to bring out the sensitivity of error to horizon T. To illustrate this point, we have included Table 3, which compares optimal value estimates obtained for $T = 10$ with 50 and 200 iterations. Plots of optimal value estimates over 200 SFP iterations for $T = 10$ are also shown in Figure 4. The choice of 200 was motivated by the fact that complexity of the standard backward recursion method of dynamic programming grows linearly with horizon (Section 5.1 of [5]); since we used 50 iterations for $T = 3$, 200 iterations for $T = 10$ seem appropriate.

Table 3: Comparison of optimal value estimates averaged over 30 independent runs with 50 and 200 iterations for Example 1 with $T = 10$.

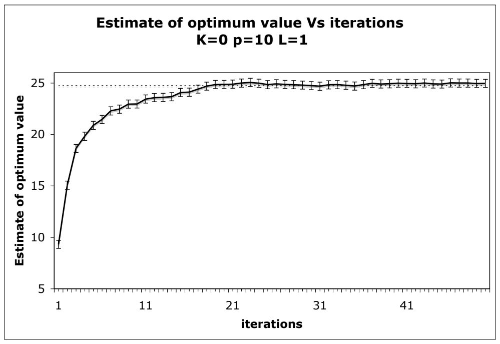
(K,p)	$V^*(s_1 = 5)$	50 iterations	200 iterations	% improvement
(0,1)	38.706	42.1640	40.7585	3.33%
(0,10)	74.7333	79.6176	77.3510	2.84%
(5,1)	41.9351	50.7621	47.6187	6.19%
(5,10)	97.038	101.505	98.5661	2.89%

5.2 Stochastic Inventory Control Example 2

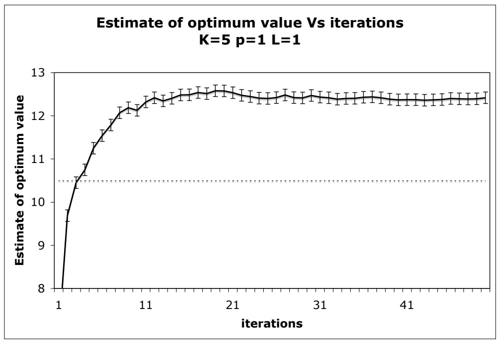
Our second example is also taken from [7] and is similar to Example 1 above except that we may now order any integer amount $\{0, 1, \ldots, 20\}$ (and so A=21). Since SFP performed slightly better with $L = 1$ for the first example, we only focus on that case for the second example. The numerical results are presented in Table 4. The second column lists the average estimate of optimal value over 30 independent runs of 5000 iterations each. As before, we include for comparison three estimates obtained by the AMS algorithm of [7] with the smallest ($N = 21$) and the largest ($N = 35$) sample



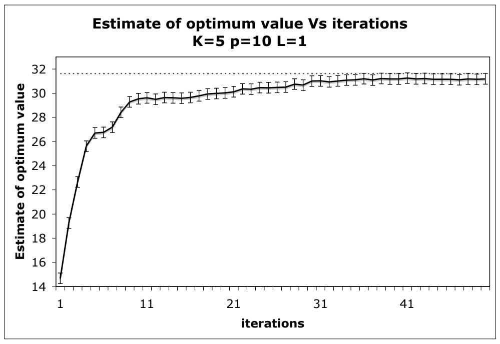
(a)



(b)

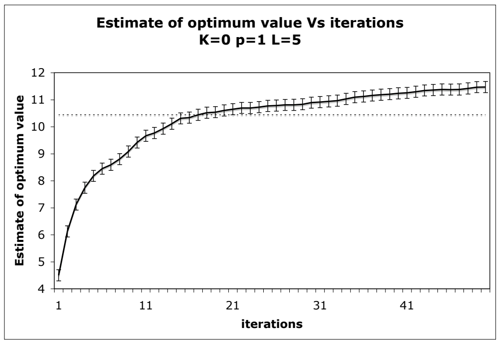


(c)

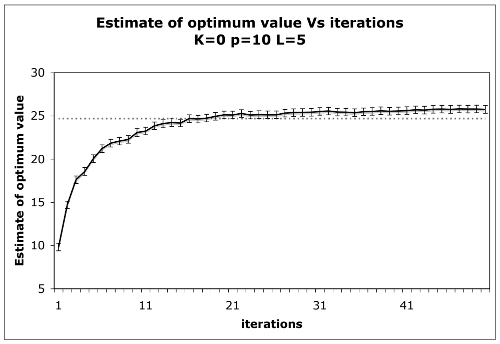


(d)

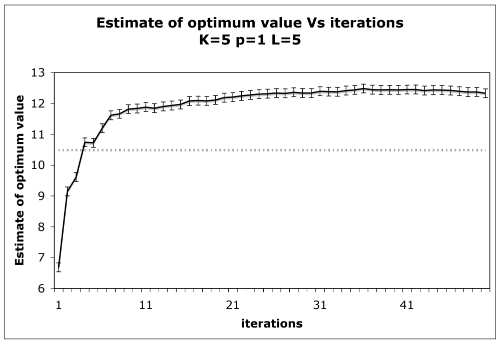
Figure 1: Evolution of the optimal value estimate with iterations for four test problems with $L = 1$ in Example 1. Each plot shows the estimates averaged over thirty independent runs, and the corresponding standard error bars. The optimal value reported in Table 1 is shown on each plot with a flat line.



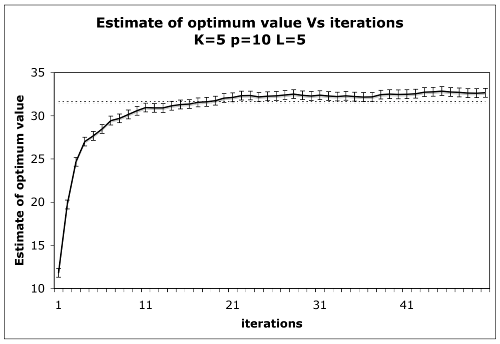
(a)



(b)

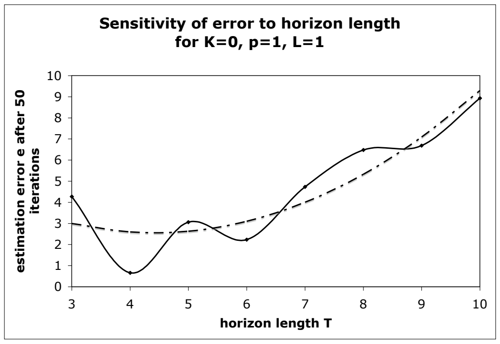


(c)

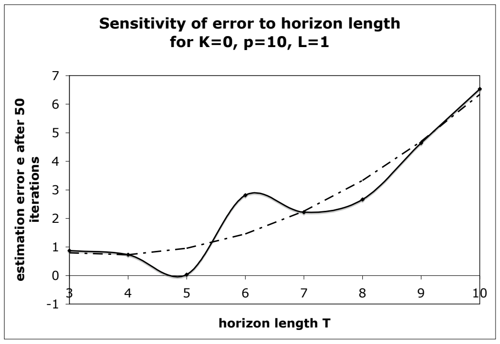


(d)

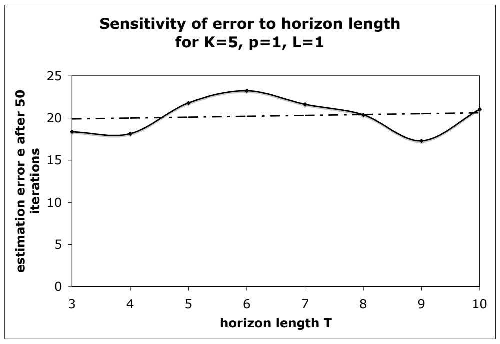
Figure 2: Evolution of the optimal value estimate with iterations for four test problems with $L = 5$ in Example 1. Each plot shows the estimates averaged over thirty independent runs, and the corresponding standard error bars. The optimal value reported in Table 1 is shown on each plot with a flat line.



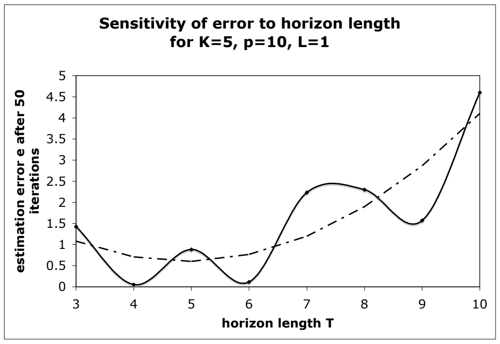
(a)



(b)

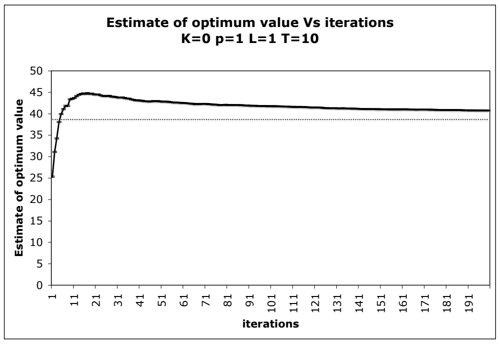


(c)

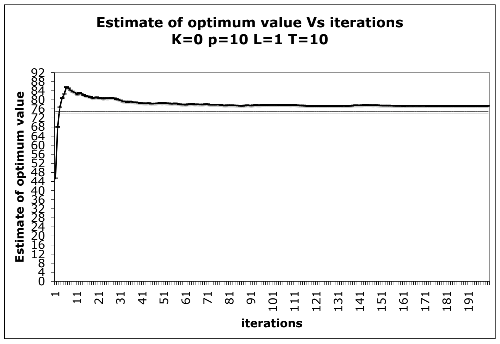


(d)

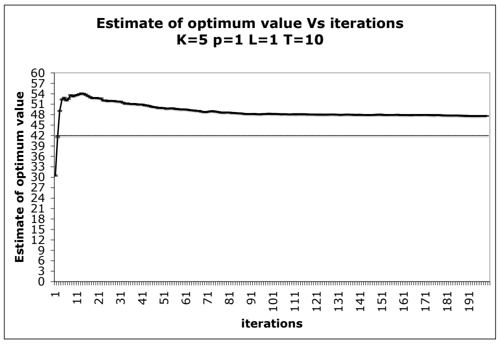
Figure 3: Sensitivity of the accuracy of our optimal value estimates to problem horizon in stochastic inventory control Example 1. The percentage relative error between optimal values and their estimates in Table 2 is given by $e = 100 \times |V(s_1 = 5) - V^*(s_1 = 5)|/V^*(s_1 = 5)$. In each plot, these relative errors are shown as black dots and are connected by a smoothed solid black line to illustrate the error trend. The dashed trend lines in each plot were obtained by fitting either a quadratic (in (a), (b), (d)) or a linear (in (c)) curve to the error data.



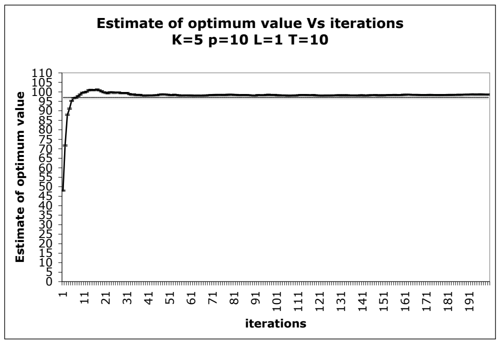
(a)



(b)



(c)



(d)

Figure 4: Evolution of the optimal value estimate with iterations for four test problems with $L = 1$ in Example 1 for $T = 10$. Each plot shows the estimates for 200 SFP iterations averaged over 30 independent runs, and the corresponding standard error bars. The optimal value is shown on each plot with a flat line.

sizes that were reported there. The convergence behavior of our average optimal value estimate with iterations is illustrated in Figure 5.

We again turn to the number of states sampled to compare computational effort of SFP and AMS. The number of states sampled by AMS is $(AN)^T = (21 \times 21)^3 =$ 85,766,121 when $N = 21$ and $(21 \times 35)^3$ =397,065,375 when $N = 35$, whereas it is $5000 \times 21 \times 12/2$ =630,000 in SFP.

Table 4 shows that, in spite of the significantly smaller number of states sampled by SFP, its optimal value estimates are considerably more accurate than those of AMS for twenty of the twenty four combinations of parameter values (K, p), estimator type and sample size, and close in the remaining four instances. The percentage relative error $\frac{100\times|V(s_1=5)-V^\star(s_1=5)|}{V^\star(s_1=5)}$ in SFP estimates across problem instances and estimator types is on average about 15 times smaller than that of AMS with $N = 21$, and about 8 times smaller with $N = 35$, even though in these two cases SFP samples only 0.74% and 0.16%, respectively, of the states sampled by AMS.

Table 4: Estimates $V(s_1 = 5)$ of $V^*(s_1 = 5)$ for stochastic inventory control Example 2 averaged over 30 independent runs of 50 iterations of SFP with $L = 1$. The three estimates reported in [7] each with sample sizes $N = 21$ and $N = 35$ are listed in the last three columns for comparison.

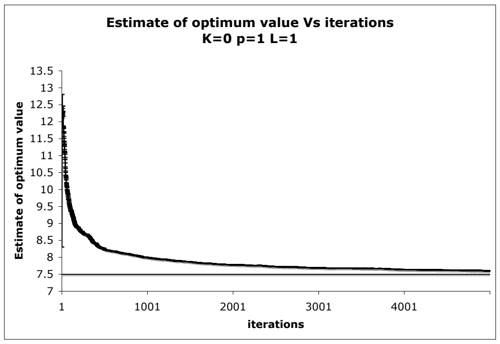
		SFP estimates	AMS estimates from [7]		
(K,p)	$V^*(s_1 = 5)$	$L = 1$	Estimate 1 N=21, N=35	Estimate 2 N=21, N=35	Estimate 3 N=21, N=35
(0,1)	7.5	7.5920	24.06, 18.82	3.12, 6.26	9.79, 6.62
(0,10)	13.5	13.5874	29.17, 26.06	6.04, 12.23	12.06, 13.69
(5,1)	10.49	12.2923	33.05, 25.33	8.73, 10.96	18.62, 11.12
(5,10)	25.785	24.6296	39.97, 36.89	17.78, 24.71	26.76, 25.51

5.3 TIC-TAC-TOE Against Nature

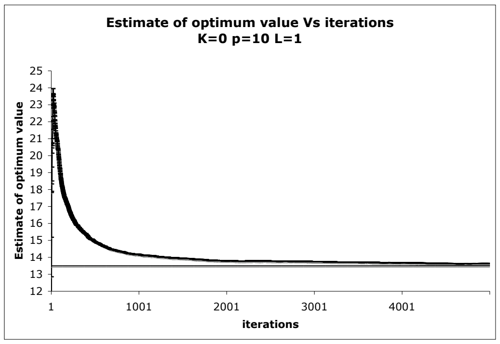
TIC-TAC-TOE is a well-known relatively simple two-player game played on a 3 by 3 square grid (called the game-board) where the two players take turns placing distinct tokens (typically an "X" and an "O") in empty squares on the grid. The first player to place three tokens in a horizontal row, a vertical column, or a diagonal wins the game. Thus the game lasts for a total of at most nine moves. It is well-known [32] that every game of TIC-TAC-TOE between two "expert players" ends in a tie whereas an expert player never loses no matter whom it is pitted against [25].

In this section we focus on TIC-TAC-TOE *against nature*, where the second player places its tokens in one of the empty squares chosen randomly. Such a player is also sometimes termed a "novice player." Thus, even when the first player employs a deterministic strategy, its game versus a novice player unfolds stochastically. One web site [25] on TIC-TAC-TOE reports that out of a thousand game experiments between an expert player and a novice player, the expert player won nine hundred and seventy eight times whereas the other twenty two games ended in a tie. Thus, roughly speaking, the probability that an expert player wins against a novice player is about 0.978 and that the game ends in a tie is about 0.022.

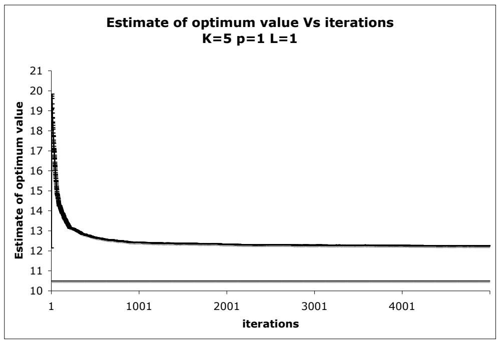
We model TIC-TAC-TOE against nature as a sequential decision problem where our decision maker corresponds to the first player playing (say) "X" who makes the first move starting with an empty game-board. The states of this problem correspond to the different possible game-board configurations. Since every square on the board can either be empty, occupied with an "X" or an "O", the total number of states is 3^9 =19,683. In fact, one can provide a more compact description



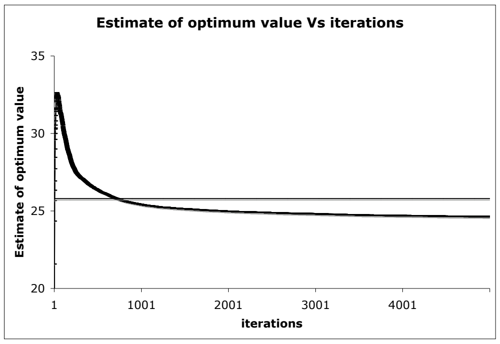
(a)



(b)



(c)

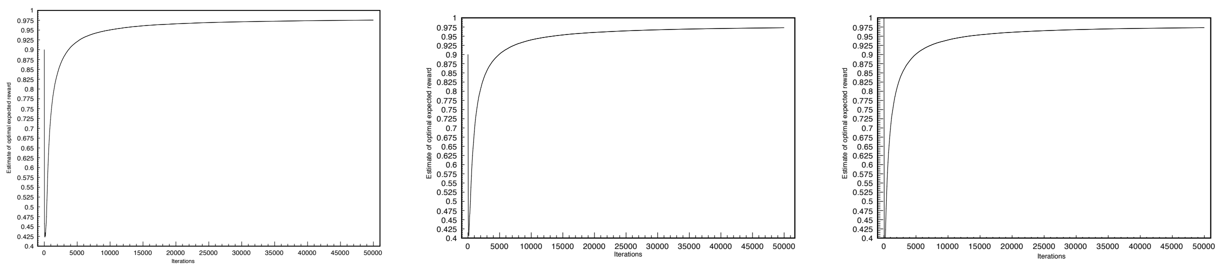


(d)

Figure 5: Evolution of the optimal value estimate with iterations for four test problems with $L = 1$ in Example 2. Each plot shows the estimates averaged over thirty independent runs, and the corresponding standard error bars. The optimal value reported in Table 4 is shown on each plot with a flat line.

of the state space of the game by taking into account rotational symmetries, but we deliberately did not incorporate this into our implementation since we wanted to evaluate how SFP performs with a larger state-space. Note that in a given state, the first player's feasible actions correspond to choosing one of the empty squares to place an "X." Thus, the number of feasible actions is at most nine in any state. Once the first player chooses an action in a specific state, this state-action pair is "sent to an oracle" that essentially returns the new game-board, or equivalently, the state reached after nature, i.e., the second player places an "O" on a randomly chosen empty square. This sequence of events continues until the game ends either in a tie or with one of the two players winning. The first player receives a reward of 1 for winning, a reward of -1 for losing and 0 for a tie. Notice that rewards are earned at game termination and there are no intermediate rewards along the way. Strictly speaking, terminal rewards do not fit into the assumptions of our analysis, but our proofs can be modified in a straightforward manner to accommodate them. Our goal is to find the *optimal expected reward* for the first player from the initial state where the game-board is empty. Note that the numbers from the web site mentioned above imply that this reward is roughly 0.978 ($0.978 \times 1 + 0.022 \times 0$) for an expert player. Thus we expect the optimal expected reward to be very close to 0.978.

We ran 10 independent trials with 50,000 iterations each of asynchronous SFP on this problem. We used $\alpha_k = (1/k)^{1/9}$ and three different L values: $L = 1$, $L = 5$ and $L = 10$. The average of our estimate of the optimal expected value over these 10 trials is plotted versus iterations in Figure 6. The average and standard deviation of estimates obtained after 50,000 iterations are shown in Table 5. Note again that our SFP procedure does not exploit symmetries or the structure of the game and does not know that the game is being played against nature but rather learns the optimal expected reward by repeated stochastic simulations leading to a relatively high number of iterations required.



(a) (b) (c)

Figure 6: Evolution of the estimate of optimal expected reward with iterations for TIC-TAC-TOE against nature with (a) $L = 1$ (b) $L = 5$ and (c) $L = 10$.

Table 5: Average and standard deviation of optimal expected reward estimates over 10 independent trials of 50,000 iterations each for TIC-TAC-TOE against nature.

L	average	standard deviation
1	0.975322	0.001299451
5	0.973054	0.00153441
10	0.97319	0.00127577

6 Discussion and Conclusions

We have presented a variant of Sampled Fictitious Play (SFP) that converges to a deterministic optimal policy while solving model-free, finite horizon stochastic dynamic programs. This convergence result is in stark contrast with the recently proposed, original version of SFP in [20], which may only converge in a very weak sense to the set of mixed strategy Nash equilibria, i.e., to the set of sub-optimal randomized policies. Our SFP variant adaptively biases the sampling mechanism towards high quality actions by endowing the players with only finite recall, and allows the players to make occasional sampling mistakes to ensure that all players will participate in the game infinitely often. Most importantly, it is computationally far less demanding than the original version in [20]. It is well-known that table look-up simulation-based methods that explicitly store state value or state-action value information, as in our SFP approach here, work well for small to moderate sized model-free problems but are unlikely to successfully tackle very large problems [3, 4]. For instance, the memory requirement for algorithms using state-action value (Q-value) information is proportional to the product of the sizes of the state and action spaces. We expect SFP to have this feature as well.

Our SFP procedure can be seen as an asynchronous version of a Q-value based [34] variant of Monte Carlo-Optimistic Policy Iteration [4, 26, 33] for model-free finite-horizon ADP [3, 9, 26, 32] (Optimistic Policy Iteration is a somewhat non-standard name used in [4] and [33], whereas in [26] it is called hybrid value/policy iteration). In the infinite horizon case, synchronous versions for such algorithms (see [33] Section 5) maintain Q-values for *each* state-action pair and generate infinite trajectories starting at *every* state-action pair using greedy actions (actions that optimize the Q-values) in every state visited. The Q-values are then updated by "smoothing" (commonly by averaging) observed cumulative costs along these trajectories. Even though these methods are only of academic interest [33] as they require generation of infinite trajectories, their analysis provides significant insights into implementable model-free ADP algorithms. In addition, in some cases, as for example where eventual absorption into a zero cost terminal state is inevitable along every trajectory, these methods are indeed implementable in practice. Interestingly, even though variants of Optimistic Policy Iteration have been known for some time, their detailed theoretical analyses have been rare. In fact, in a list of important open questions in reinforcement learning, Sutton [31] had posed whether synchronous Q-function based Monte Carlo-Optimistic Policy Iteration converges to optimality. This question was recently resolved in the affirmative by Tsitsiklis [33] using monotonicity properties of the dynamic programming operator. Also, non-convergence of *asynchronous* Monte Carlo-Optimistic Policy Iteration is well-known [33] for infinite-horizon problems. We were not able to find a rigorous proof for convergence of asynchronous Monte Carlo-Optimistic Policy Iteration methods for the model-free finite-horizon case in the literature. We have shown that strong convergence results for our model-free finite-horizon Monte Carlo algorithm are easy to prove using backward induction, and the strong law of large numbers, even in the asynchronous case. Part of this simplicity stems from the way in which Q-values are updated in our SFP procedure — update is performed for every feasible action in a selected state rather than only for the action currently estimated to be optimal (see the asynchronous SFP pseudo-code and discussion in Section 3).

As noted in the literature [4, 32], a key to ensuring convergence to optimality in simulation-based asynchronous ADP algorithms without significantly sacrificing computational performance is to bias the underlying action sampling procedure toward good actions and hence states, and yet sample all states infinitely often, that is, to properly balance exploitation with exploration. Most asynchronous techniques take an ad-hoc approach toward this issue [26]. Recently developed algorithms that attempt to carefully tradeoff exploitation versus exploration for simulation-based methods for model-free finite-horizon stochastic dynamic programs are surveyed in [8] with expanded details appearing in [9]. Our approach in this paper is close in spirit to two Multi-stage Adaptive Sampling Algorithms discussed in the second chapter of [9]. Of these, the Upper Confidence Bound (UCB) algorithm from [7] exploits the theory of multi-armed bandit problems [1] to sample actions to minimize expected regret (see Section 2.1.4 in [9]), whereas the Pursuit Learning Automata (PLA) technique samples actions from a probability distribution biased toward an estimate of an optimal action essentially implementing a recursive extension of the Pursuit Algorithm [28, 32]. This probability distribution is iteratively updated explicitly through an update formula that increases the probability of sampling an action estimated to be optimal, that is, a "greedy action", by a fraction β (see Equation 2.12 in [32]). This "pursuit" of a greedy action motivated the name "Pursuit Algorithm." In our SFP approach as well, actions are sampled from a probability distribution — the one induced by the history of recent plays. This probability distribution is adaptively updated, not by using an explicit formula as in PLA, but rather implicitly and naturally as the history of plays evolves with iterations. Whereas the estimates of optimal expected value in PLA converge in probability (Theorem 2.9 page 50 in [9]), we are able to obtain almost sure convergence results because we do not have exogenous noise in the action sampling mechanism employed in the best response process. Specifically, we proved in Theorem 4.1 that our SFP variant converges to a deterministic optimal policy with probability one. Corollary 4.6 showed that the value estimates generated by SFP converge with probability one to the optimal expected multi-period reward. In contrast, the UCB technique converges in mean (Theorem 2.3 page 29 in [9]).

Extension of results in this paper to infinite-horizon model-free problems is a natural direction for future research. However, since it is not possible to sample an infinite path in practice, an implementable procedure should approximate the infinite-horizon expected rewards with some appropriately selected finite-horizon truncations. This will introduce an additional hurdle in proving convergence to optimality.

In addition, the fundamental distinguishing features of our variant of SFP — finite memory, action samples of size one, and occasional mistakes in sampling — appear to help, both in simplifying convergence proofs, as well as enhancing computational performance as compared to the original version of SFP in [20]. Therefore, we will focus on investigating theoretical and empirical properties of this variant when applied to combinatorial optimization problems that are not modeled as dynamic programs. We have taken initial steps in this direction, obtaining encouraging numerical results comparable to Genetic Algorithms [11], on multi-dimensional knapsack problems [17].

Researchers have recently proposed "payoff-based" dynamics for learning equilibria in multiplayer games [23]. In contrast to "action-based" mechanisms such as FP, or AP, these approaches do not require a best response computation. For example, in the so-called *safe experimentation dynamics* [23], players simply maintain a record of the best utility value they obtained in the past, and the corresponding action they played at the time. These are called the baseline utility and action, respectively. In each iteration k, each player samples its baseline action with probability ϵ_k, and with probability $1 - \epsilon_k$, samples an action randomly. If the utility a particular player receives for the joint action profile sampled by all players is higher than its baseline utility, that player updates its baseline utility and action. It is easy to prove (Theorem 3.1 in [23]) that this dynamic process converges almost surely to a pure strategy optimal Nash equilibrium for all games of identical interests under a standard sufficient condition on ϵ_k similar to our condition on α_k in Lemma 4.2. In the future, it would be interesting to compare the computational performance of such payoff-based processes on dynamic programs with the action-based approach in this paper.

Acknowledgments

This research was funded in part by the National Science Foundation under grants CMMI-0422752, CCF-0830380, and CCF-0830092. Archis Ghate appreciates summer support from the University of Washington.

References

[1] Auer, P., Cesa-Bianchi, N., Fisher, P., Finite-time analysis of the multiarmed bandit problem, *Machine Learning*, 47, 235-256, 2002.

[2] Baumert, S., Cheng, S. F., Ghate, A. V., Reaume, D., Sharma, D., Smith, R. L., Joint optimization of capital investment, revenue management, and production planning in complex manufacturing systems, Technical Report 05-05, Industrial and Operations Engineering, University of Michigan, Ann Arbor, 2005.

[3] Bertsekas, D. P., Dynamic programming and optimal control, volumes 1 and 2, Athena Scientific, Belmont, MA, USA, 1995.

[4] Bertsekas, D. P., and Tsitsiklis, J. N., Neuro-dynamic programming, Athena Scientific, Belmont, MA, USA,1996.

[5] Blondel, V. D., and Tsitsiklis, J. N., A survey of computational complexity results in systems and control, *Automatica*, 36, 2000.

[6] Brown, G. W., Iterative solution of games by fictitious play, in *Activity Analysis of Production and Allocation*, Wiley, New York, NY, USA, 1951.

[7] Chang, H. S., Fu, M.C., Hu, J., and Marcus, S. I., An adaptive sampling algorithm for solving Markov decision processes, *Operations Researh*, 53 (1), 126-139, 2005.

[8] Chang, H. S., Fu, M. C., Hu, J., Marcus, S. I., A survey of some simulation-based algorithms for Markov decision processes, *Communications in Information and Systems*, 7 (1), 59-92, 2007.

[9] Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I., Simulation-based algorithms for Markov decision processes, Springer, London, UK, 2007.

[10] Cheng, S. F., Epelman, M. A., Smith, R. L., CoSIGN: a parallel algorithm for coordinated traffic signal control, *IEEE Transactions on Intelligent Transportation Systems*, 7 (4), 551-564, 2007.

[11] Chu, P. C., and Beasley, J. E., A genetic algorithm for the multi-dimensional knapsack problem, *Journal of Heuristics*, 4, 63-86, 1998.

[12] Feller, W., An introduction to probability theory and its applications, John Wiley and Sons, New York, NY, USA, 1970.

[13] Fudenberg, D., and Tirole, J., Game Theory, MIT Press, Cambridge, MA, USA, 1991.

[14] Garcia, A., Reaume, D., Smith, R.L., Fictitious play for finding system optimal routings in dynamic traffic networks, *Transportation Research B, Methods*, 34 (2), 2000.

[15] Garcia, A. Patek, S. D., and Sinha, K., A decentralized approach to discrete optimization via simulation: application to network flows, *Operations Research*, 55 (4), 2007.

[16] Garcia, A. and Campos, E., Game theoretic approach to efficient power management in sensor networks, *Operations Research*, 56 (3), 2008.

[17] Ghate, A. V., Game theory, Markov chains and infinite programming: three paradigms for optimization of complex systems, Ph.D. Thesis, Industrial and Operations Engineering, The University of Michigan, Ann Arbor, 2006.

[18] Gershwin, S. B., Manufacturing systems engineering, Prentice-Hall, Englewood Cliffs, NJ, USA, 1994.

[19] Gigerenzer, G., and Selten, R., Bounded rationality, MIT Press, Cambridge, MA, USA, 2002.

[20] Lambert, T.J., Epelman, M.A., and Smith, R.L., A fictitious play approach to large-scale optimization, *Operations Research*, 53 (3), pp. 477-489, 2005.

[21] Lambert, T. J., and H. Wang, Fictitious play approach to a mobile unit situation awareness problem, Technical Report, University of Michigan, Ann Arbor, 2003.

[22] Marden, J. R., Arslan, G., and Shamma, J. S., Joint strategy fictitious play with inertia for potential games, 44th IEEE Conference on Decision and Control, December 2005.

[23] Marden, J. R., Young, H. P., Arslan, G., and Shamma, J. S., Payoff-based dynamics for multiplayer weakly acyclic games, *SIAM Journal of Control and Optimization*, 48 (1), 373-396.

[24] Monderer, D., Shapley, L., Fictitious play property for games with identical interests, *Journal of Economic Theory*, 68, 258-265, 1996.

[25] Ostermiller, S., http://ostermiller.org/tictactoeexpert.html, last accessed August 3, 2010.

[26] Powell, W., Approximate dynamic programming: solving the curses of dimensionality, John Wiley and Sons, New York, NY, USA, 2007.

[27] Puterman, M., Markov Decision Processes, John Wiley and Sons, New York, NY, USA, 1994.

[28] Rajaraman, K. and Sastry, P. S., Finite time analysis of the pursuit algorithm for learning automata, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26 (4), 590-598, 1996.

[29] Robinson, J., An iterative method of solving a game, *Annals of Mathematics*, 54, 296-301, 1951.

[30] Stout, W., Almost sure convergence, Academic Press, New York, NY, 1974.

[31] Sutton, R. S. Open theoretical questions in reinforcement learning. In Fischer, P., and Simon, H.U. (Eds.), Proceedings of the Fourth European Conference on Computational Learning Theory (Proceedings EuroCOLT99), pages 11-17, 1999. Springer- Verlag. ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/sutton-99.ps

[32] Sutton, R., and Barto, A., Reinforcement learning: an introduction, MIT Press, Cambridge, MA, USA, 1998.

[33] Tsitsiklis, J. N., On the convergence of optimistic policy iteration, *Journal of Machine Learning Research*, 3, 59 - 72, 2002.

[34] Watkins, C., and Dayan, P., Technical Note: Q-Learning, *Machine Learning*, 8, 279-292, 1992.

[35] Young, H. P., Evolution of conventions, *Econometrica*, 61 (1), 57-84, 1993.