Asynchronous Modified Policy Iteration with Single-sided Updates

Satinder P. Singh
Department of Brain and Cognitive Sciences
MIT
Cambridge, MA 02139
singh@psyche.mit.edu

Vijaykumar Gullapalli
Computer Science Department
University of Massachusetts
Amherst, MA 01003
vijay@cs.umass.edu

December 20, 1993

Abstract

We present a new algorithm for solving Markov decision problems that extends the modified policy iteration algorithm of Puterman and Shin [6] in two important ways: 1) The new algorithm is asynchronous in that it allows the values of states to be updated in arbitrary order, and it does not need to consider all actions in each state while updating the policy. 2) The new algorithm converges under more general initial conditions than those required by modified policy iteration. Specifically, the set of initial policy–value function pairs for which our algorithm guarantees convergence is a strict superset of the set for which modified policy iteration converges. This generalization was obtained by making a simple and easily implementable change to the policy evaluation operator used in updating the value function. Both the asynchronous nature of our algorithm and its convergence under more general conditions expand the range of problems to which our algorithm can be applied.

Keywords: Markov Decision Problems, Asynchronous, Single-sided.

The research presented here was done while the first author was a graduate student at the Department of Computer Science at the University of Massachusetts. This material is based upon work supported by funding provided to Prof. Andrew G. Barto by the AFOSR, Bolling AFB, under Grant AFOSR-F49620-93-1-0269 and by the NSF under Grant ECS-92-14866.

1 Introduction

The problem of finding optimal policies in finite Markovian decision processes (MDPs) can be reduced to the problem of solving a finite system of non-linear equations known as the Bellman optimality equation (Bellman [2], see also Bertsekas [3]). There are two classical iterative dynamic programming (DP) methods for solving the Bellman equation: policy iteration (Howard [5]), and value iteration (see Ross [7]). Puterman and Shin [6] have shown that it is more appropriate to think of these two classical methods as two extremes of a continuum of iterative methods, consisting of what they call *modified policy iteration* algorithms.

Modified policy iteration is a synchronous algorithm, i.e., it updates information about every state of the MDP in each iteration. Puterman and Shin proved that under certain initial conditions, any synchronous modified policy iteration method is guaranteed to yield an optimal policy. Recently, Williams and Baird [9] derived an asynchronous algorithm that can be viewed as a form of asynchronous modified policy iteration (Barto [1]). Williams and Baird proved convergence of their algorithm, hereafter called the W&B algorithm, under initial conditions similar to those specified by Puterman and Shin (P&S). In this paper, we present a new algorithm that is also an example of asynchronous modified policy iteration, but converges under less restrictive initial conditions than both the P&S and the W&B algorithms.

2 Solving Markovian Decision Processes

Consider an MDP with a finite state set X, a finite set of actions A, a payoff function $R : X \times A \to \mathbb{R}$, and a state transition function $P : X \times A \times X \to [0, 1]$. A stationary policy, $\pi : X \to A$, is a function that assigns an action to each state. The optimal decision problem requires finding a policy that maximizes some given objective functional defined over policy space. A commonly used functional is the expected value of the infinite-horizon sum of discounted payoffs as a function of the start state: $V^\pi(x) = E\{\sum_{t=0}^{\infty} \gamma^t R^{a_t}(x_t) | x_0 = x\}$, where $V^\pi(x)$ is the *value* of state x under policy π, $0 \leq \gamma < 1$ is the discount factor, x_t and a_t are the state and action at step t, $a_t = \pi(x_t)$, and $R^a(x)$ is the payoff received on executing action a in state x. Here, the expectation is over the random state transitions and the possibly random payoffs received at each time step.

Let $\mathcal{P}$ be the set of stationary policies. An optimal policy[1] $\pi^* \in \mathcal{P}$ is such that $\forall x \in X$, and for all $\pi \in \mathcal{P}$, $V^{\pi^*}(x) \geq V^\pi(x)$. The value function for an optimal policy is denoted V^*. The optimal value function V^* satisfies the following Bellman equation:

$$V(x) = \max_{a \in A} \left[R^a(x) + \gamma \sum_{y \in X} P^a(x, y) V(y) \right]$$

where $P^a(x, y)$ is the probability of a transition to state y when action a is executed in

[1]For finite MDPs it is known that there is a stationary optimal policy (e.g., Ross [7]).

state x. An optimal policy π^* can be found from V^* as follows:

$$\pi^*(x) \;=\; \text{argmax}_{a \in A} \left[R^a(x) + \gamma \sum_{y \in X} P^a(x,y) V^*(y) \right],$$

where ties between actions can be broken arbitrarily.

3 Asynchronous Policy Iteration

The algorithms defined in this paper are asynchronous iterative algorithms of the following general form:

$$(V_{k+1}, \pi_{k+1}) \;=\; U_k(V_k, \pi_k)$$

where (V_k, π_k) is the k^{th} estimate of (V^*, π^*), and U_k is the asynchronous update operator applied at iteration k. We now define the relevant asynchronous update operators that can be used in an iteration.

3.1 Asynchronous Update Operators

For ease of exposition, define

$$Q^V(x,a) = R^a(x) + \gamma \sum_{y \in X} P^a(x,y) V(y), \tag{1}$$

where $Q^V(x,a)$ is the expected "value" of taking action a in state x when the resulting states are evaluated using the value function V. We define the following asynchronous update operators:

1. A policy evaluation operator, $B_x(V_k, \pi_k)$, uses the current value function V_k and the current policy π_k to update the value of state x. If $U_k = B_x$, then

$$\begin{aligned} V_{k+1}(i) &= \begin{cases} Q^{V_k}(x, \pi_k(x)) & \text{if } i = x \\ V_k(i) & \text{otherwise, and} \end{cases} \\ \pi_{k+1} &= \pi_k. \end{aligned}$$

2. A *single-sided* policy evaluation operator, $T_x(V_k, \pi_k)$, also uses the current value function V_k and the current policy π_k to update the value of state x as follows:

$$\begin{aligned} V_{k+1}(i) &= \begin{cases} \max(Q^{V_k}(x, \pi_k(x)),\ V_k(x)) & \text{if } i = x \\ V_k(i) & \text{otherwise, and} \end{cases} \\ \pi_{k+1} &= \pi_k. \end{aligned}$$

   The policy evaluation operator T_x is called single-sided because it never causes the value of a state to decrease.

3. A *single-action* policy improvement operator, $L_x^a(V_k, \pi_k)$, uses the current value function V_k and the current policy π_k to update the policy for state x as follows:

$$\begin{aligned} V_{k+1} &= V_k, \text{and} \\ \pi_{k+1}(y) &= \begin{cases} a & \text{if } y = x \text{ AND } Q^{V_k}(x,a) > Q^{V_k}(x, \pi_k(x)) \\ \pi_k(y) & \text{otherwise.} \end{cases} \end{aligned}$$

The operator L_x^a is termed single-action because it only considers action a in updating the policy for state x.

4. A policy improvement operator, $L_x(V_k, \pi_k)$, that corresponds to the sequential application of the operators $L_x^{a_1} L_x^{a_2} \ldots L_x^{a_{|A|}}$. Therefore, $(V_{k+1}, \pi_{k+1}) = L_x(V_k, \pi_k)$ implies that

$$\begin{aligned} V_{k+1} &= V_k, \text{and} \\ \pi_{k+1}(y) &= \begin{cases} \arg\max_{a \in A} Q^{V_k}(x,a) & \text{if } y = x, \\ \pi_k(y) & \text{otherwise.} \end{cases} \end{aligned}$$

The operator $L_x(V, \pi)$ considers all possible actions in state x and updates $\pi(x)$ to be the optimal action with respect to V.

3.2 Asynchronous Single-sided Policy Iteration (ASPI)

The ASPI algorithm is defined by the iteration

$$(V_{k+1}, \pi_{k+1}) = U_k(V_k, \pi_k), \quad k = 0, 1, 2, \ldots, \tag{2}$$

where $U_k \in \{T_x \mid x \in X\} \bigcup \{L_x^a \mid x \in X, a \in A\}$. Many other DP algorithms are also of the form given in Equation 2; the differences lie in the choice of operators U_k. For example,

- The W&B algorithm: $U_k \in \{B_x \mid x \in X\} \bigcup \{L_x^a \mid x \in X, a \in A\}$.
- Asynchronous value iteration: $U_k \in \{B_x L_x \mid x \in X\}$.
- The order-m Gauss-Sidel P&S algorithm: $U_k = B^m L, \ m \geq 1$, where $B = B_{x_1} B_{x_2} \ldots B_{x_{|X|}}$ and $L = L_{x_1} L_{x_2} \ldots L_{x_{|X|}}$.
- Gauss-Sidel value iteration algorithm: $U_k = BL$.
- The policy iteration algorithm: $U_k = B^\infty L$.

3.3 Convergence Results

Let the set of *non-overestimating* value functions, $\{V \in \mathbb{R}^{|X|} | V \leq V^*\}$, be denoted $\mathcal{V}$. The convergence analysis of the ASPI algorithm presented in this paper will be based on the assumption that the initial value-policy pair $(V_0, \pi_0) \in (\mathcal{V} \times \mathcal{P})$. No constraints are placed on the initial policy π_0. In contrast with this initial condition, to prove convergence

of their algorithm, Puterman and Shin require that (V_0, π_0) be such that $V_0 \in \{V \in \mathbb{R}^{|X|} | \max_\pi (R^\pi + \gamma [P]^\pi V) \geq 0\}$, which is a strict subset of $\mathcal{V}$. Similarly, the initial condition required for convergence of the W&B algorithm is that for all $x \in X$, $Q^{V_0}(x, \pi_0(x)) \geq V_0(x)$, which is again a strict subset of $(\mathcal{V} \times \mathcal{P})$.

We begin by proving the following Lemmas.

Lemma 1: For the ASPI algorithm, $V_{k+1} \geq V_k$ for all k.

Proof: By the definitions of operators T_x and L_x^a, the value of any state is never decreased.
Q.E.D.

Lemma 2: If (V_0, π_0) is such that $V_0 \in \mathcal{V}$, then the ASPI algorithm ensures that $V_k \in \mathcal{V}$ for all k.

Proof: We will prove Lemma 2 by induction. Clearly, by assumption, $V_0 \in \mathcal{V}$. Assume that $V_k \in \mathcal{V}$ for all $k \leq m$. There are only two kinds of operations possible in iteration $m+1$:

1. Operator $U_m = L_x^a$ for some arbitrary state-action pair in $X \times A$. Then, $V_{m+1} = V_m \leq V^*$.

2. Operator $U_m = T_x$ for some arbitrary state $x \in X$. Then U_m will only impact $V_{m+1}(x)$.

$$\begin{aligned} V_{m+1}(x) &= \max(Q^{V_m}(x, \pi_m(x)), V_m(x)) \\ &= \max([R^{\pi_m(x)}(x) + \gamma \sum_{y \in X} P^{\pi_m(x)}(x,y) V_m(y)], V_m(x)) \\ &\leq \max([R^{\pi_m(x)}(x) + \gamma \sum_{y \in X} P^{\pi_m(x)}(x,y) V^*(y)], V_m(x)) \\ &\leq \max([R^{\pi^*(x)}(x) + \gamma \sum_{y \in X} P^{\pi^*(x)}(x,y) V^*(y)], V_m(x)) \\ &\leq V^*(x). \end{aligned}$$

   Hence $V_{m+1} \in \mathcal{V}$.

Q.E.D.

Theorem 1: Given an initial value-policy pair (V_0, π_0), such that $V_0 \in \mathcal{V}$, the ASPI algorithm $(V_{k+1}, \pi_{k+1}) = U_k(V_k, \pi_k)$ converges to (V^*, π^*) under the following conditions:
*A*1) For each $x \in X$, T_x appears in $\{U_k\}$ infinitely often, and
*A*2) For each $(x, a) \in X \times A$, L_x^a appears in $\{U_k\}$ infinitely often.

Proof: The formal proof of this theorem is presented in Appendix A.[2] Intuitively, the proof is based on the following observation. It is possible to partition the sequence $\{U_k\}$ into disjoint subsequences of finite length, such that each subsequence satisfies the following property: $\forall x \in X$, there is a sub-subsequence that is an arbitrary permutation of $L_x^{a_1} L_x^{a_2} \ldots L_x^{a_{|A|}}$ followed by a T_x. Each such subsequence leads to a contraction in the max-norm of the error in the approximation to V^*. Lemma 2 and the contraction mapping theorem can then be used to infer convergence to (V^*, π^*).

Corollary 1: Define the operator $T \doteq T_{x_1} T_{x_2} \ldots T_{x_{|X|}}$ and operator $L \doteq L_{x_1} L_{x_2} \ldots L_{x_{|X|}}$. Let the operator $U_k = T^m L$, where $m > 1$. Then, given a starting value-policy pair (V_0, π_0), such that $V_0 \in \mathcal{V}$, the iterative algorithm $(V_{k+1}, \pi_{k+1}) = U_k(V_k, \pi_k)$ converges to (V^*, π^*).

4 Discussion

The ASPI algorithm, like the W&B algorithm, is more "finely" asynchronous than conventional asynchronous DP algorithms such as asynchronous value iteration (AVI) in two ways:

1. ASPI allows arbitrary sequences of policy evaluation and policy improvement operators as long as they satisfy conditions $A1$ and $A2$. AVI, on the other hand, is more coarsely asynchronous because it does not separate the two functions of policy improvement and policy evaluation. In effect, AVI iterates a single operator that does policy improvement followed immediately by one step of policy evaluation[3].

2. Because AVI uses the policy improvement operator, it has to consider all actions in the state being updated. ASPI on the other hand can sample a single action in each state to perform single-action policy improvements.

The only difference between ASPI and the W&B algorithm, and between the synchronous algorithm presented in Corollary 1 and the P&S algorithm, is in the use of the single-sided policy evaluation operator. This small, easily implementable change allows convergence under less restrictive initial conditions.

A. Proof of Theorem 1

Throughout this section we will use the shorthand $(V_{l+h}, \pi_{l+h}) = \{U_k\}_l^{l+h-1}(V_l, \pi_l)$ for

$$(V_{l+h}, \pi_{l+h}) \;=\; U_{l+h-1} U_{l+h-2} \ldots U_l (V_l, \pi_l).$$

[2]The operators T_x and L_x^a defined in Section 3.1 and the initial condition stated in this section are based on the assumption that the objective functional was to be maximized. If the objective functional is to be minimized, the following changes have to be made to obtain a convergence result corresponding to Theorem 1: $\mathcal{V}$ has to be the set of non-underestimating value functions, the max in the definition of T_x will have to be replaced by a min, and the argmax in the definition of L_x^a will have to be replaced by an argmin.

[3]Of course, the policy evaluation operator used by AVI is not single-sided.

For ease of exposition, define the *identity* operator $I(V_k, \pi_k)$ that makes no changes at all, i.e.

$$(V_{k+1}, \pi_{k+1}) = I(V_k, \pi_k) = (V_k, \pi_k).$$

Further, define the composite operator $C_x = T_x L_x$ for all $x \in X$.

Fact 1: Consider a sequence of operators $\{U_k\}_l^{l+h-1}$ such that for $l \leq k \leq (l+h-1)$, $U_k \in \{C_x \mid x \in X\}$, and for all $x \in X$, $C_x \in \{U_k\}_l^{l+h-1}$. Then if $V_l \in \mathcal{V}$, $||V_{l+h} - V^*||_\infty \leq \gamma ||V_l - V^*||_\infty$.

Proof: This is a simple extension of a result in Bertsekas and Tsitsiklis [4].

Fact 2: Consider the iteration $(V'_{k+1}, \pi_{k+1}) = U_k(V'_k, \pi_k)$, where $U_k \in \{C_x \mid x \in X\}$. If $V'_0 \in \mathcal{V}$, and for all $x \in X$, $C_x \in \{U_k\}$ infinitely often, then $\lim_{k\to\infty} V'_k = V^*$.

Proof: This result follows from Fact 1 and the contraction mapping theorem. Note that the iteration defined in Fact 2 is a single-sided version of asynchronous value iteration that updates both a value function as well as a policy.
Q.E.D.

Fact 3: Consider $(V_{k+1}, \pi_{k+1}) = C_x(V_k, \pi_k)$ and $(V'_{k+1}, \pi'_{k+1}) = C_x(V'_k, \pi'_k)$. If $V^* \geq V_k \geq V'_k$, then for all $\pi_k, \pi'_k \in \mathcal{P}$, $V_{k+1} \geq V'_{k+1}$.

Proof:

$$\begin{aligned} V'_{k+1}(x) &= R^{\pi'_{k+1}}(x) + \gamma \sum_{y\in X} P^{\pi'_{k+1}}(x,y) V'_k(y) \\ &= \max_{a\in A} \left[R^a(x) + \gamma \sum_{y\in X} P^a(x,y) V'_k(y) \right] \end{aligned}$$

because $\pi'_{k+1}(x)$ is the optimal action with respect to V'_k. Similarly,

$$\begin{aligned} V_{k+1}(x) &= \max_{a\in A} \left[R^a(x) + \gamma \sum_{y\in X} P^a(x,y) V_k(y) \right] \\ &\geq R^{\pi'_{k+1}}(x) + \gamma \sum_{y\in X} P^{\pi'_{k+1}}(x,y) V_k(y) \\ &\geq R^{\pi'_{k+1}}(x) + \gamma \sum_{y\in X} P^{\pi'_{k+1}}(x,y) V'_k(y) \\ &\geq V'_{k+1}(x) \end{aligned}$$

Q.E.D.

Lemma 3: Consider a sequence of operators $\{U_k\}_l^{l+h-1}$ such that for some arbitrary state x, for all k, $l \leq k \leq l+h-2$, $U_k \in \{L_x^a \mid x \in X, a \in A\}$, and for all $a \in A$, $L_x^a \in \{U_k\}_l^{l+h-2}$,

and $U_{l+h-1} = T_x$. Let $V_l \in \mathcal{V}$, and let $(V, \pi) = C_x(V_l, \pi_l) = T_x L_x(V_l, \pi_l)$. Then, $V_{l+h} = V$.

Proof: Let $(V_{l+h-1}, \pi_{l+h-1}) = \{U_k\}_l^{l+h-2}(V_l, \pi_l)$. Then for all k, $l \leq k \leq (l+h-1)$, $V_k = V_l$, and since $\forall a \in A$, $L_x^a \in \{U_k\}_l^{l+h-2}$, $\pi_{l+h-1}(x)$ will be an optimal action with respect to V_l. Therefore, $(V_{l+h}, \pi_{l+h}) = \{U_k\}_l^{l+h-1}(V_l, \pi_l) = T_x(V_{l+h-1}, \pi_{l+h-1}) = T_x(V_l, \pi_{l+h-1}) = (V, \pi_{l+h})$.
Q.E.D.

Define $W(x)$ to be the set of *finite* length sequences of operators that satisfy the following properties: each element $\{w_k\}_0^{h-1} \in W(x)$ has a subsequence $\{w_k\}_0^d$, where $d < h-1$, and $\forall a \in A$, $L_x^a \in \{w_k\}_0^d$, and $w_{h-1} = T(x)$. Note that for each state $x \in X$, there is a separate set $W(x)$.

Lemma 4: In ASPI, for any arbitrary state x, consider a subsequence of operators such that $\{U_k\}_l^{l+h-1} \in W(x)$. Let $(V, \pi) = C_x(V_l, \pi_l)$. If $V_l \in \mathcal{V}$, then $V_{l+h} \geq V$.

Proof: By definition, any element of $W(x)$ involves applying each operator in the set $\{L_x^a | a \in A\}$ followed by the operator T_x. Although other operators may also be applied intermixed with these, intermediate applications of L_y^a where $y \neq x$ will not affect the policy for state x and intermediate applications of any T_y can only increase the value function. This observation, combined with Lemma 3, constitutes a proof.
Q.E.D.

Define W to be a set of finite length sequences of operators that satisfy the following property: each element of W contains *disjoint* subsequences, such that $\forall x \in X$ at least one distinct subsequence is in $W(x)$.

Lemma 5: Consider any sequence $\{U_k\}_l^{l+h-1} \in W$. If $V_l \in \mathcal{V}$, then $||V_{l+h} - V^*||_\infty \leq \gamma ||V_l - V^*||_\infty$.

Proof: From Lemma 4 and Facts 1 and 3 it can be seen that applying any sequence of operators that is an element of W results in a contraction.

Theorem 1: Given an initial value-policy pair (V_0, π_0), such that $V_0 \in \mathcal{V}$, the ASPI algorithm $(V_{k+1}, \pi_{k+1}) = U_k(V_k, \pi_k)$ converges to (V^*, π^*) under the following conditions:
*A*1) For each $x \in X$, T_x appears in $\{U_k\}$ infinitely often, and
*A*2) For each $(x, a) \in X \times A$, L_x^a appears in $\{U_k\}$ infinitely often.

Proof: The infinite sequence $\{U_k\}$ has an infinity of disjoint subsequences that are elements of W. The result that $\lim_{k\to\infty}(V_k, \pi_k) = (V^*, \pi_\infty)$ follows from Lemma 5 and the contraction mapping theorem. Moreover, it is known that there exists an $\epsilon > 0$ such that if $||V - V^*||_\infty \leq \epsilon$ then the optimal policy with respect to V is also optimal with respect to V^* (e.g., Singh [8]). Because the sequence $\{V_k\}$ is non-decreasing and $V_k \leq V^*$, it can

be concluded that $\pi_\infty \in \{\pi^*\}$.
Q.E.D.

References

[1] A.G. Barto. personal communication.

[2] R.E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[3] D.P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[4] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[5] R. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA, 1960.

[6] M.L. Puterman and M.C. Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11), July 1978.

[7] S. Ross. *Introduction to Stochastic Dynamic Programming.* Academic Press, New York, 1983.

[8] S. P. Singh. *Learning to Solve Markovian Decision Processes.* PhD thesis, Department of Computer Science, University of Massachusetts, 1993.

[9] R. J. Williams and L. C. Baird. Analysis of some incremental variants of policy iteration: First steps toward understanding actor-critic learning systems. Technical Report NU-CCS-93-11, Northeastern University, College of Computer Science, Boston, MA 02115, September 1993.