Distributed Asynchronous Policy Iteration in Dynamic Programming

Dimitri P. Bertsekas and Huizhen Yu

Abstract— We consider the distributed solution of dynamic programming (DP) problems by policy iteration. We envision a network of processors, each updating asynchronously a local policy and a local cost function, defined on a portion of the state space. The computed values are communicated asynchronously between processors and are used to perform the local policy and cost updates. The natural algorithm of this type can fail even under favorable circumstances, as shown by Williams and Baird [WiB93]. We propose an alternative and almost as simple algorithm, which converges to the optimum under the most general conditions, including asynchronous updating by multiple processors using outdated local cost functions of other processors.

I. INTRODUCTION

The field of distributed computation is experiencing renewed interest in the context of systems of multiple communicating processors/agents, each with its own autonomous computation capability. The processors take responsibility for local computation/decision making, while exchanging information with neighboring processors. In such a system it is natural for the processors to operate asynchronously, possibly updating and communicating their decision/computation variables at different rates. Furthermore, the communicated results may be out-of-date for a variety of reasons, which we loosely refer to in this paper as communication delays.

Generally, convergent distributed iterative asynchronous algorithms are classified in totally and partially asynchronous (cf. Chapters 6 and 7 of the book [BeT89]). In the former, there is no bound on the communication delays, while in the latter there must be a bound (which may be unknown). These two types of asynchronous algorithms differ not only in their assumptions, but also, more fundamentally, in the convergence mechanisms and corresponding lines of analysis. The algorithms of the present paper are totally asynchronous versions of the classical policy iteration algorithm for dynamic programming (DP), possibly in its modified form where policy evaluation is performed with a finite number of value iterations (see e.g., Puterman [Put94]).

The idea of asynchronous computation originated in the 60s, when it was seen as a means to alleviate the "communication penalty" in parallel computation and data network

Dimitri Bertsekas is with the Dept. of Electr. Engineering and Comp. Science, M.I.T., Cambridge, Mass., 02139. dimitrib@mit.edu

Huizhen Yu is with the Dept. of Computer Science, Univ. of Helsinki, Finland. janey.yu@cs.helsinki.fi

communication. This is the inefficiency that results from synchronizing iterations across multiple processors that may operate at different speeds and may be subject to inter-communication delays. Chazan and Miranker [ChM69] first analyzed the convergence of totally asynchronous Jacobi-type algorithms for iterative solution of linear systems of equations involving sup-norm contractions (they attributed the idea of asynchronous iterative computation to Rosenfeld). Related algorithms were subsequently studied by several authors for nonlinear equations involving sup-norm contractions (Baudet [Bau78], Bertsekas [Ber83], Miellou [Mie75], Robert [Rob76], El Tarazi [ElT82], Spiteri [Spi86], and Tsitsiklis [Tsi87]), and for equations involving a monotonicity structure, such as DP/value iteration (Bertsekas [Ber82]), and Jacobi-type dual network optimization (Bertsekas and ElBaz [BeE87], Bertsekas and Eckstein [BeE88], Tseng, Bertsekas, and Tsitsiklis [TBT90], Bertsekas and Castañon [BeC91], El Baz, et. al. [ESM96], and Beraldi and Guerriero [BeG97]).

In this paper we focus on algorithms for the distributed solution of the DP/Bellman's equation $J = TJ$, where J represents the cost function of the problem and T is the mapping associated with a value iteration. An example is the infinite horizon α-discounted stochastic optimal control problem, involving a cost per stage $g(x_k, u_k, w_k)$ and a discrete-time system equation $x_{k+1} = f(x_k, u_k, w_k)$, where x_k and u_k are the state and control at time k, and w_k is a random vector. In this case J is a function of the state x and T has the form

$$(TJ)(x) = \min_{u \in U(x)} E\Big\{ g(x, u, w) + \alpha J\big(f(x, u, w)\big) \Big\}, \ (1.1)$$

where $U(x)$ is a set of admissible controls at state x, and the expected value is taken over w (see e.g., [Ber07], Chapter 1).

We consider a distributed computation framework involving a partition of X into disjoint nonempty subsets, and a network of processors, each updating their corresponding local components of J, while using the local components of other processors, which are communicated with some "delay." Note that each processor knows the (complete) state x within the local subset of the state space, but does not know fully the values of the cost vector outside the local subset (these are communicated by other processors). This is different from much of the literature on multi-agent system models, where the state may consist of the Cartesian product of the "local" states of the agents, none of whom may know the complete/global state of the system.

Several earlier works are relevant to the present paper. Two of these are the asynchronous distributed shortest path and DP algorithm of [Ber82], and the general convergence

theorem of [Ber83] for deterministic totally asynchronous iterations, which also served as the foundation for the treatment of totally asynchronous iterations in the book by Bertsekas and Tsitsiklis ([BeT89], Chapter 6). The rate of convergence analysis given in [Ber89], Section 6.3.5 is also relevant to our algorithms, but we have not focused on this subject here. Our earlier paper (Bertsekas and Yu [BeY10]) is the original source for several of the ideas of the present paper and served as its starting point (see the subsequent discussion for relations between the present paper and [BeY10]).

To provide an overview of our algorithms for finding a fixed point of a mapping T, which maps functions J of $x \in X$ into functions TJ of x, let us consider a partition of X into disjoint nonempty subsets $X_1, \ldots, X_n$, and let J be partitioned as

$$J = (J_1, \ldots, J_n),$$

where J_i is the restriction of J on the set X_i. Each processor $i = 1, \ldots, n$, of the distributed computing system is in charge of updating the corresponding component J_i. Let T_i be the corresponding component of T, so that the equation $J = TJ$ can equivalently be written as the system of n equations

$$J_i(x) = \big(T_i(J_1, \ldots, J_n)\big)(x), \quad \forall \, x \in X_i, \; i = 1, \ldots, n.$$

Then in a (synchronous) distributed value iteration algorithm, each processor i updates J_i at iteration t according to

$$J_i^{t+1}(x) = \big(T_i(J_1^t, \ldots, J_n^t)\big)(x), \quad \forall \, x \in X_i. \tag{1.2}$$

In an asynchronous value iteration algorithm, processor i updates J_i only for t in a selected subset $\overline{\mathbf{T}}_i$ of iterations, and with components J_j supplied by other processors with communication "delays" $t - \tau_{ij}(t)$, i.e., for all $x \in X_i$,

$$J_i^{t+1}(x) = \big(T_i(J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)})\big)(x), \quad \forall \, x \in X_i. \tag{1.3}$$

At times $t \notin \overline{\mathbf{T}}_i$, processor i leaves $J_i(x)$ unchanged, i.e., $J_i^{t+1}(x) = J_i^t(x)$.

The convergence of this algorithm was first studied in [Ber82], and was also discussed as a special case of a general asynchronous convergence framework given in [Ber83]. For a very broad class of DP problems, the conditions for asynchronous convergence are essentially that:

(1) The set of times $\overline{\mathbf{T}}_i$ at which processor i updates J_i is infinite, for each i.
(2) $\lim_{t \to \infty} \tau_{ij}(t) = \infty$ for all $i, j = 1, \ldots, n$.
(3) The mapping T is monotone (i.e., $TJ \leq TJ'$ if $J \leq J'$), and has a unique fixed point within a range of functions of interest. [In our notation, inequalities involving functions are meant to be by component, e.g., $J \leq J'$ means $J(x) \leq J'(x)$ for all $x \in X$.]

The key to the convergence proof of [Ber82] is the monotonicity property of T, although a sup-norm contraction property of T, when present [as for example in discounted stochastic optimal control problems where T has the form

(1.1)] was mentioned as a sufficient condition for asynchronous convergence in [Ber82] and [Ber83]. The monotonicity property also plays a key role in the analysis of the present paper.

In this paper we study distributed asynchronous policy iteration algorithms that have comparable properties to the value iteration algorithm (1.3) under the conditions (1)-(3) above. In a natural asynchronous version of the policy iteration algorithm, the processors collectively maintain and update an estimate J^t of the optimal cost function, and an estimate μ^t of an optimal policy. The local portions of J^t and μ^t of processor i are denoted J_i^t and μ_i^t, respectively, i.e., $J_i^t(x) = J^t(x)$ and $\mu_i^t(x) = \mu^t(x)$ for all $x \in X_i$.

To illustrate, let us focus on discounted stochastic optimal control problems where T has the form (1.1). For each processor i, there are two disjoint subsets of times $\overline{\mathbf{T}}_i$ and $\mathbf{T}_i$. At each time t:

(a) If $t \in \overline{\mathbf{T}}_i$, processor i performs a *policy improvement* iteration, where the local policy is set to one attaining the corresponding minimum in the value iteration (1.3), i.e., for all $x \in X_i$,

$$\mu_i^{t+1}(x) \in \arg\min_{u \in U(x)} E\Big\{g(x, u, w) + \alpha J^{i,t}\big(f(x, u, w)\big)\Big\}, \tag{1.4}$$

where $J^{i,t} = \big(J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)}\big)$. This local policy improvement is done simultaneously with the local cost update/value iteration (1.3).

(b) If $t \in \mathbf{T}_i$, processor i performs a *policy evaluation* iteration at x, where the local cost vector J_i is updated, using the current (local) policy μ_i and "delayed" cost components of other processors, i.e., for all $x \in X_i$,

$$J_i^{t+1}(x) = \begin{cases} \big(T_{i,\mu_i^t}(J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)})\big)(x) & \text{if } t \in \mathbf{T}_i, \\ J_i^t(x) & \text{if } t \notin \mathbf{T}_i, \end{cases} \tag{1.5}$$

where

$$(T_{i,\mu_i}J)(x) = E\Big\{g(x, \mu_i(x), w) + \alpha J\big(f(x, \mu_i(x), w)\big)\Big\}.$$

(c) If $t \notin \overline{\mathbf{T}}_i \cup \mathbf{T}_i$, processor i leaves J_i and μ_i unchanged, i.e., $J_i^{t+1}(x) = J_i^t(x)$, and $\mu_i^{t+1}(x) = \mu_i^t(x)$ for all $x \in X_i$.

Note that if a policy improvement is done by all processors at every iteration via Eq. (1.4), the algorithm is equivalent to the synchronous value iteration algorithm (1.2). An interesting implementation is when the set of policy improvement times $\overline{\mathbf{T}}_i$ has a lot fewer elements than the set of policy evaluation times $\mathbf{T}_i$. This results in the typical advantage that policy iteration-type methods (or modified policy iteration in the terminology of Puterman [Put94]) hold over value iteration methods (the more expensive policy improvement using a computation of T_i is done less frequently than a policy evaluation using T_{i,μ_i}). Note, however, that even when implemented as a nondistributed algorithm, modified policy iteration may require that (J^0, μ^0) satisfies the condition $J^0 \geq T_{\mu^0} J^0$ for convergence. This restriction is undesirable, particularly in a distributed setting, or when a favorable

initial condition is known from solution of a related problem. For discounted DP with a standard Jacobi-like policy evaluation this restriction is not needed (see [Ber07], p. 88). It may, however, be needed for Gauss-Seidel variants (see [Put94], Section 6.5), or for modified policy iteration methods applied to discounted semi-Markov problems.

A distributed algorithm of the form (1.3)-(1.5), with more asynchronous character than the Gauss-Seidel implementation of [Put94], was studied by Williams and Baird [WiB93] for discounted finite-state Markovian decision problems (MDP). In their analytical framework there are no communication delays [i.e., $\tau_{ij}(t) = t$], and only one processor updates at a time (i.e., $t \in \overline{\mathbf{T}}_i$ or $t \in \mathbf{T}_i$ for only one i), but the order of processor updates, and their type (policy evaluation versus policy improvement) can be arbitrary. Convergence of the method (1.3)-(1.5) to J^*, the optimal cost function/fixed point of T, was proved in [WiB93] under some assumptions, chief of which is the condition $J^0 \geq T_{\mu^0} J^0$ (the same as in [Put94]). This analysis may also be found in [BeT96] (Prop. 2.5) and [Ber07] (Prop. 1.3.5). Convergence was also shown in [WiB93] for some variants of iteration (1.5) under the assumption $J^0 \geq J^*$. Moreover it was shown in [WiB93], with several deterministic counterexamples, how the algorithm can oscillate/fail if the condition $J^0 \geq T_{\mu^0} J^0$ is not satisfied. Thus failure is due to making policy updates one processor at a time in a disorderly fashion rather than due to stochastic effects or communication delays.

The purpose of this paper is to propose new asynchronous policy iteration algorithms, which allow for delays, and improve on the existing methods in two ways:

(a) *They embody a mechanism that precludes the convergence failure described above, without requiring the condition $J^0 \geq T_{\mu^0} J^0$.* In particular, our algorithms update the local policy of processor i by using the minimization (1.4), but modify the policy evaluation Eq. (1.5) in a simple way so that convergence is guaranteed. The idea is that convergence may fail because through iteration (1.5), $J_i^{t+1}(x)$ may be increased more than appropriate relative to the value iterate $\left(T_i(J_1^t, \ldots, J_n^t)\right)(x)$ [cf. Eqs. (1.3) and (1.5)]. Stated in simple terms, our algorithms include a mechanism that senses when this may be dangerous, and use a more conservative update instead.

(b) *They are more general than the one based on Eqs. (1.3)-(1.5)* because they can be applied beyond discounted finite-state MDP. This is done through the use of the abstract DP model introduced in the paper by Bertsekas [Ber77], and used extensively in the book by Bertsekas and Shreve [BeS78]. This model contains as special cases several of the standard DP problems, including discounted and stochastic shortest path MDP, Q-learning, deterministic shortest path, multiplicative/exponential cost, minimax, DP models with aggregation, etc.

The mechanism to preclude convergence failure [cf. (a) above] is one of main ideas of our earlier paper [BeY10], where several closely related distributed asynchronous policy

iteration algorithms were proposed. A key idea of [BeY10] was to modify the policy evaluation phase of policy iteration for Q-factors in discounted MDP so that it solves an optimal stopping problem. This idea has another important use: it can provide the basis for exploration enhancement, which is essential in policy iteration with Q-factor approximation (see e.g., [BeT96], [SuB98]). Moreover [BeY10] proposed the use of compact Q-factor representations, and derived associated error bounds. Some of our algorithms admit stochastic iterative implementations, which may be analyzed using asynchronous stochastic approximation theory (Tsitsiklis [Tsi94]). Several such algorithms were proposed and analyzed in [BeY10], using compact Q-factor representations and (approximate) solution of an optimal stopping problem with the algorithm of Tsitsiklis and Van Roy [TsV99].

The paper is organized as follows. In Section II we introduce the abstract DP model, and in Section III we state our assumptions. In Section IV we give our algorithm and convergence result, and in Section V we discuss an algorithmic variation.

II. AN ABSTRACT DP MODEL

We adopt the general abstract DP framework of [Ber77] and [BeS78]. Let X be a set of states, U be a set of controls, and for each $x \in X$, let $U(x) \subset U$ be a subset of controls that are feasible at state x. Functions of the form $\mu : X \mapsto U$ with $\mu(x) \in U(x)$ for all $x \in X$ are referred to as *policies*, and the set of all policies is denoted by $\mathcal{M}$.

Let F be the set of extended real-valued functions $J : X \mapsto [-\infty, \infty]$. For any two functions $J, J' \in F$, we write

$$J \leq J' \quad \text{if} \quad J(x) \leq J'(x), \quad \forall\, x \in X,$$

$$J = J' \quad \text{if} \quad J(x) = J'(x), \quad \forall\, x \in X.$$

Let $H : X \times U \times F \mapsto [-\infty, \infty]$ be a given mapping, and consider the mapping $T : F \mapsto F$ defined by

$$(TJ)(x) = \min_{u \in U(x)} H(x, u, J), \qquad \forall\, x \in X, \qquad (2.1)$$

and for each policy $\mu \in \mathcal{M}$, the mapping $T_\mu : F \mapsto F$ defined by

$$(T_\mu J)(x) = H\big(x, \mu(x), J\big), \qquad \forall\, x \in X. \qquad (2.2)$$

We assume that the minimum in Eq. (2.1) is attained for all $x \in X$ and $J \in F$. This is necessary since we will be dealing with policy iteration-type algorithms. Given a subset $\overline{F}$ of functions within F, the problem is to find a function $J^* \in \overline{F}$ such that

$$J^*(x) = \min_{u \in U(x)} H(x, u, J^*), \qquad \forall\, x \in X,$$

i.e., find within $\overline{F}$ a fixed point J^* of T. We also want to find a policy μ^* such that $T_{\mu^*} J^* = T J^*$.

We consider a partition of X into disjoint nonempty subsets $X_1, \ldots, X_n$. For each $J \in F$, let $J_i : X_i \mapsto [-\infty, \infty]$ be the restriction of J on X_i. Similarly, let H_i

be the ith component of H, so that the equation $J = TJ$ can be equivalently written as the system of n equations

$$J_i(x) = \min_{u \in U(x)} H_i(x, u, J_1, \ldots, J_n), \quad \forall \ x \in X_i,$$

for $i = 1, \ldots, n$. For each $\mu \in \mathcal{M}$, let $\mu_i : X_i \mapsto U$ and $T_{i,\mu_i} J$ be the restrictions of μ and $T_\mu J$ on X_i, respectively, so the components of $T_\mu J$ are defined by

$$(T_{i,\mu_i} J)(x) = H_i\big(x, \mu_i(x), J_1, \ldots, J_n\big), \quad \forall \ x \in X_i, \quad (2.3)$$

for $i = 1, \ldots, n$. We introduce a network of n processors, which asynchronously update local estimates of J and μ. In particular, processor i updates J_i and μ_i, and communicates J_i (or a compact/basis function representation of J_i) to the other processors $j \neq i$.

We give a few examples, and we refer to [Ber77], [BeS78], [Ber82] for additional examples.

Example 2.1: (Discounted Markovian Decision Problems) Consider an α-discounted MDP involving states $x = 1, \ldots, n$, controls $u \in U(x)$ at state x, transition probabilities $p_{xy}(u)$, and cost per stage $g(x, u, y)$. For

$$H(x, u, J) = \sum_{y=1}^{n} p_{xy}(u)\big(g(x, u, y) + \alpha J(y)\big),$$

the equation $J = TJ$, with T given by Eq. (2.1), is Bellman's equation for the MDP. Here each set X_i consists of a single state i, and processor i updates $J(i)$, and communicates the result to the other processors. It is also possible to consider models where processors update the costs of multiple states, but the nature of our asynchronous computation model is such that there is no loss of generality in assuming that the number of processors is equal to the number of states, as long as the number of states is finite (we create duplicate processors if necessary). □

Example 2.2: (Deterministic and Stochastic Shortest Path Problems) Consider a classical deterministic shortest path problem involving a graph of n nodes $x = 1, \ldots, n$, plus a destination d, an arc length a_{xy} for each arc (x, y), and the mapping

$$H(x, y, J) = \begin{cases} a_{xy} + J(y) & \text{if } y \neq d, \\ a_{xd} & \text{if } y = d. \end{cases}$$

Then the equation $J = TJ$, with T given by Eq. (2.1), is Bellman's equation for the shortest distance $J(x)$ from node x to node d. A generalization is a mapping of the form

$$H(x, u, J) = p_{xd}(u)g(x, u, d) + \sum_{y=1}^{n} p_{xy}(u)\big(g(x, u, y) + J(y)\big),$$

which corresponds to a stochastic shortest path problem ([Ber07], Chapter 2), and includes as a special case stochastic finite-horizon, finite-state DP problems. □

Example 2.3: (Discounted Semi-Markov Problems) With x, y, and u as in Example 2.1, consider the mapping

$$H(x, u, J) = G(x, u) + \sum_{y=1}^{n} m_{xy}(u) J(y)$$

where G is some function representing cost per stage, and $m_{xy}(u)$ are nonnegative numbers with $\sum_{y=1}^{n} m_{xy}(u) < 1$ for all $x \in X$ and $u \in U(x)$. The equation $J = TJ$, with T given by Eq. (2.1), can be viewed as Bellman's equation for a continuous-time semi-Markov decision problem, after it is converted into an equivalent discrete-time problem (see e.g., [Ber07], Section 5.3). □

Example 2.4: (Minimax Problems) For the MDP notation of Example 2.1 and an additional antagonistic player choosing a variable w from a set $W(x, u)$, consider the mapping

$$H(x, u, J) = \max_{w \in W(x,u)} \sum_{y=1}^{n} p_{xy}(u, w)\big(g(x, u, w, y) + \alpha J(y)\big).$$

Then the equation $J = TJ$, with T given by Eq. (2.1), is Bellman's equation for a minimax MDP. □

Example 2.5: (Distributed Policy Iteration with Cost Function Approximation) In large-scale problems, it may be inconvenient for the processors to exchange their local cost functions $J_i(x)$ for all $x \in X_i$. It is thus interesting to introduce cost function approximation, and an algorithmic framework where each processor updates J_i, while using approximate cost functions/compact representations of J_j, $j \neq i$, communicated by other processors.

For instance, in the context of a finite-state discounted MDP, suppose that each processor i maintains/updates a (local) cost $J_i(x)$ for every state $x \in X_i$, and an aggregate cost

$$R_i = \sum_{x \in X_i} d_{ix} J_i(x), \quad (2.4)$$

where $\{d_{ix} \mid x \in X_i\}$ is a probability distribution over X_i. We generically denote by J and R the vectors with components $J(x)$, $x \in X$, and R_i, $i = 1, \ldots, n$, respectively. Consider the mapping H_i defined for all $i = 1, \ldots, n$, $x \in X_i$, $u \in U(x)$, and J, by

$$\begin{aligned} H_i(x, u, J) &= \sum_{y \in X} p_{xy}(u)g(x, u, y) + \alpha \sum_{y \in X_i} p_{xy}(u)J(y) \\ &\quad + \alpha \sum_{y \notin X_i} p_{xy}(u)R_{j(y)} \\ &= \sum_{y \in X} p_{xy}(u)g(x, u, y) + \alpha \sum_{y \in X_i} p_{xy}(u)J(y) \\ &\quad + \alpha \sum_{y \notin X_i} p_{xy}(u) \sum_{z \in X_{j(y)}} d_{j(y)z} J_{j(y)}(z), \end{aligned}$$

$$(2.5)$$

and where for each original system state y, we denote by $j(y)$ the subset to which y belongs [i.e., $y \in X_{j(y)}$]. Then the solution of the equation $J = TJ$, with T given by Eq. (2.1) is an approximate/aggregation-based Bellman's equation for the MDP. Alternatively, we may consider a mapping

$$H_i(x, u, J, R)$$

that involves both J and the corresponding aggregate vector R, as defined by Eq. (2.4). □

III. Assumptions for Asynchronous Convergence

We introduce the assumptions for our algorithm to be introduced in Section IV.

Assumption 3.1: (Monotonicity)
(a) *The mapping H is monotone within $\overline{F}$ in the sense that*

$$H(x, u, J) \le H(x, u, J'),$$

for all $x \in X$, $u \in U(x)$, and $J, J' \in \overline{F}$ such that $J \le J'$.

(b) *The mapping T has a unique fixed point J^* within $\overline{F}$.*

(c) *There exist two functions $\underline{J}$ and $\overline{J}$ in $\overline{F}$ such that all functions $J \in F$ with $\underline{J} \le J \le \overline{J}$ belong to $\overline{F}$, and we have*

$$\underline{J} \le T\underline{J} \le T\overline{J} \le \overline{J}.$$

Furthermore,

$$\lim_{k \to \infty} (T^k \underline{J})(x) = \lim_{k \to \infty} (T^k \overline{J})(x) = J^*(x), \quad \forall\, x \in X.$$

In our algorithm, we assume the following regarding the initial estimates J_i^0 held at processors $i = 1, \ldots, n$:

Assumption 3.2: (Initial Conditions) *The initial functions $J^0 = (J_1^0, \ldots, J_n^0)$ satisfy $\underline{J} \le J^0 \le \overline{J}$.*

Note that Assumptions 3.1 and 3.2 (for any initial J^0) are satisfied for discounted DP problems with bounded cost per stage, and for stochastic shortest path (SSP) models (see the discussions in [Ber77], [BeS78], [Ber82], [Ber07]). For example in the discounted MDP Example 2.1, one may take for all $x \in X$,

$$\underline{J}(x) = -\beta, \qquad \overline{J}(x) = \beta, \tag{3.1}$$

where β is a sufficiently large number. For the case of SSP models (assuming all policies are proper), see the construction of the proof of Prop. 2.2.3 of [Ber07].

In our algorithms, the processors asynchronously update local estimates of J and μ using *policy improvement* iterations similar to Eq. (1.4), and *policy evaluation* iterations similar to Eq. (1.5). For each i, there are two disjoint subsets of times $\mathbf{T}_i, \overline{\mathbf{T}}_i \subset \{0, 1, \ldots\}$, corresponding to policy evaluation and policy improvement iterations, respectively. At the times $t \in \mathbf{T}_i \cup \overline{\mathbf{T}}_i$, the local cost function J_i^t of processor i is updated using "delayed" local costs $J_j^{\tau_{ij}(t)}$ of other processors $j \ne i$, where $0 \le \tau_{ij}(t) \le t$, and at the times $t \in \overline{\mathbf{T}}_i$, the local policy μ_i^t is also updated. For various choices of $\overline{\mathbf{T}}_i$ and $\mathbf{T}_i$, the algorithm takes the character of value iteration (when $\overline{\mathbf{T}}_i = \{0, 1, \ldots\}$), and policy iteration (when $\mathbf{T}_i$ contains a large number of time indexes between successive elements of $\overline{\mathbf{T}}_i$). We view $t - \tau_{ij}(t)$ as a "communication delay." We consider $\tau_{ij}(t)$ as being defined for all t [even though the algorithm uses $\tau_{ij}(t)$ only for $t \in \mathbf{T}_i \cup \overline{\mathbf{T}}_i$], and we assume the following.

Assumption 3.3: (Continuous Updating and Information Renewal) *For each $i = 1, \ldots, n$, the set $\overline{\mathbf{T}}_i$ is infinite, and*

$$\lim_{t \to \infty} \tau_{ij}(t) = \infty, \qquad \forall\, j = 1, \ldots, n.$$

Assuming that $\overline{\mathbf{T}}_i$ is infinite is essential for proving any kind of convergence result about our algorithms. The condition $\tau_{ij}(t) \to \infty$ guarantees that outdated information about the processor updates will eventually be purged from the multi-processor system. It would also be natural to assume that $\tau_{ij}(t)$ is monotonically increasing with t (i.e., the processors use increasingly up-to-date information), but this assumption is not necessary for our analysis.

IV. Distributed Asynchronous Policy Iteration

In our first policy iteration method, the processors collectively maintain and update functions J^t, V^t, and a policy μ^t. We assume that the initial V^0 satisfies $\underline{J} \le V^0 \le \overline{J}$ (for example we may use $V^0 = J^0$). The initial μ^0 is arbitrary. At each time t and for each processor i:

(a) If $t \in \overline{\mathbf{T}}_i$, processor i does a *policy improvement* iteration, i.e., sets for all $x \in X_i$,

$$\mu_i^{t+1}(x) = \arg\min_{u \in U(x)} H_i\big(x, u, J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)}\big), \tag{4.1}$$

$$J_i^{t+1}(x) = \min_{u \in U(x)} H_i\big(x, u, J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)}\big), \tag{4.2}$$

and records the value of the minimum in Eqs. (4.1)-(4.2) in the function $V_i : X_i \mapsto [-\infty, \infty]$:

$$V_i^{t+1}(x) = J_i^{t+1}(x). \tag{4.3}$$

(b) If $t \in \mathbf{T}_i$, processor i does a *policy evaluation* iteration, i.e., sets for all $x \in X_i$,

$$J_i^{t+1}(x) = \min\left\{ V_i^t(x), \Big(T_{i,\mu_i^t}\big(J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)}\big)\Big)(x) \right\}, \tag{4.4}$$

and leaves V_i and μ_i unchanged, i.e., $V_i^{t+1}(x) = V_i^t(x)$ and $\mu_i^{t+1}(x) = \mu_i^t(x)$ for all $x \in X_i$.

(c) If $t \notin \overline{\mathbf{T}}_i \cup \mathbf{T}_i$, processor i leaves J_i, V_i, and μ_i unchanged, i.e., $J_i^{t+1}(x) = J_i^t(x)$, $V_i^{t+1}(x) = V_i^t(x)$, and $\mu_i^{t+1}(x) = \mu_i^t(x)$ for all $x \in X_i$.

Note the difference between Eqs. (1.5) and (4.4): the latter restricts $J_i^{t+1}(x)$ not to exceed $V_i^t(x)$. This modification provides the convergence mechanism that precludes the difficulties associated with the Williams and Baird counterexamples (cf. the related remarks in the introductory section).

Proposition 4.1: *Under Assumptions 3.1-3.3 we have for all $i = 1, \ldots, n$, and $x \in X_i$,*

$$\lim_{t \to \infty} J_i^t(x) = \lim_{t \to \infty} V_i^t(x) = J^*(x).$$

Proof: The proof makes use of the functions $W_i^t : X_i \mapsto [-\infty, \infty]$ defined for all t, $i = 1$, and $x \in X_i$, by

$$W_i^{t+1}(x) = \min_{u \in U(x)} H_i\left(x, u, J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)}\right). \tag{4.5}$$

For $t = 0$, we define $W_i^0 = J_i^0$. From Eqs. (2.3), (4.2), (4.4), and (4.5), we have

$$\min\left\{V_i^t(x), W_i^{t+1}(x)\right\} \le J_i^{t+1}(x) \le V_i^t(x),$$
$$\forall\, x \in X_i,\ t \in \mathbf{T}_i, \tag{4.6}$$

$$J_i^{t+1}(x) = V_i^{t+1}(x), \qquad \forall \, x \in X_i, \, t \in \overline{\mathbf{T}}_i. \qquad (4.7)$$

The preceding two relations play an important role in the proof: it is shown that $W_i^t(x)$ and $V_i^t(x)$ converge to $J^*(x)$, so Eqs. (4.6)-(4.7) guarantee the convergence of J^t to J^*. In particular, we will show that for each $k \geq 0$, there exists an integer $t_k \geq 0$, such that for all $i = 1, \ldots, n$, we have

$$(T^k \underline{J})(x) \leq J_i^t(x) \leq (T^k \overline{J})(x), \quad \forall \, x \in X_i, \text{ and } t \geq t_k, \qquad (4.8)$$

$$(T^k \underline{J})(x) \leq V_i^t(x) \leq (T^k \overline{J})(x), \quad \forall \, x \in X_i, \text{ and } t \geq t_k, \qquad (4.9)$$

$$(T^k \underline{J})(x) \leq W_i^t(x) \leq (T^k \overline{J})(x), \quad \forall \, x \in X_i, \text{ and } t \geq t_k. \qquad (4.10)$$

In view of Assumption 3.1(c), the desired result will follow. Simultaneously with Eqs. (4.8)-(4.10), we will show that for all $i, j = 1, \ldots, n$, we have

$$(T^k \underline{J})(x) \leq J_i^{\tau_{ji}(t)}(x) \leq (T^k \overline{J})(x), \quad \forall \, x \in X_i, t \geq t_k. \qquad (4.11)$$

The proof of Eqs. (4.8)-(4.11) will be by induction on k.

We first show that Eqs. (4.8)-(4.11) hold for $k = 0$. Indeed letting $t_0 = 0$, we see that Eqs. (4.8)-(4.10) hold for $t = 0$ because of the initial condition Assumption 3.2 and the definition $W_i^0 = J_i^0$. By induction on t, they hold also for $t > 0$: suppose they hold for all $t \leq \bar{t}$, then for $t = \bar{t} + 1$, Eqs. (4.9) and (4.10) hold because the update formulas (4.3) and (4.5) for V^t and W^t, respectively, maintain the inequalities $\underline{J} \leq V^t \leq \overline{J}$ and $\underline{J} \leq W^t \leq \overline{J}$ in view of Assumption 3.1(c). From Eqs. (4.6)-(4.7) it follows that for $t = \bar{t} + 1$, Eq. (4.8) holds, so Eqs. (4.11) also holds. This shows that Eqs. (4.8)-(4.10) hold for $t \geq 0$.

Assuming that Eqs. (4.8)-(4.11) hold for some k, we will show that they hold with k replaced by $k + 1$. For every $i = 1, \ldots, n$, let $\bar{t}_k(i)$ be the first integer $t > t_k$ such that $t \in \overline{\mathbf{T}}_i$ (we use here Assumption 3.3). Then from the update formulas (4.3) and (4.5) for V^t and W^t, respectively, Assumption 3.1, and Eqs. (4.11), we have

$$(T^{k+1} \underline{J})(x) \leq V_i^t(x) \leq (T^{k+1} \overline{J})(x),$$
$$\forall \, x \in X_i, \text{ and } t > \bar{t}_k(i), \qquad (4.12)$$

$$(T^{k+1} \underline{J})(x) \leq W_i^t(x) \leq (T^{k+1} \overline{J})(x),$$
$$\forall \, x \in X_i, \text{ and } t > \bar{t}_k(i). \qquad (4.13)$$

Thus, from Eqs. (4.6)-(4.7) it follows that

$$(T^{k+1} \underline{J})(x) \leq J_i^t(x) \leq (T^{k+1} \overline{J})(x),$$
$$\forall \, x \in X_i, \text{ and } t > \bar{t}_k(i). \qquad (4.14)$$

Finally, using again Assumption 3.3, let t_{k+1} be the first integer such that for all $t \geq t_{k+1}$,

$$\tau_{ij}(t) > \max_{\ell=1,\ldots,n} \bar{t}_k(\ell), \qquad \forall \, i, j = 1, \ldots, n.$$

Then, from Eq. (4.14), we have for all $i, j = 1, \ldots, n$,

$$(T^{k+1} \underline{J})(x) \leq J_i^{\tau_{ji}(t)}(x) \leq (T^{k+1} \overline{J})(x),$$
$$\forall \, x \in X_i, \text{ and } t \geq t_{k+1}. \qquad (4.15)$$

Combining the fact $t_{k+1} > \max_{i=1,\ldots,n} \bar{t}_k(i)$ and Eqs. (4.12)-(4.15), we see that Eqs. (4.8)-(4.11) hold with k replaced by $k + 1$, and the induction is complete. **Q.E.D.**

Let us note that the preceding proof does not depend on the choice of the local policies μ_i used in the update (4.4); any choice of policy will work. On the other hand, the update of μ_i based on the minimization of Eq. (4.1) is natural and consistent with the principles of policy iteration, and comes at no additional cost since the minimization must be carried out anyway to update V_i using Eqs. (4.2) and (4.3).

V. AN ALGORITHMIC VARIATION

A potential inefficiency in the algorithm arises when we often have

$$V_i^t(x) < \left(T_{i,\mu_i^t} \big(J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)} \big) \right)(x) \qquad (5.1)$$

in Eq. (4.4), in which case $J_i^{t+1}(x)$ is set to $V_i^t(x)$ and the policy evaluation iteration is "wasted." This can be inefficient, particularly when V_i^t approaches its limit J_i^* from lower values (some value iteration and modified policy iteration algorithms, such as the Bellman-Ford algorithm for shortest paths, generally tend to work slowly, when the solution is approached from lower values, but the inefficiency noted can exacerbate the difficulty). We present an algorithmic variation that modifies appropriately Eq. (4.4), using interpolation with a stepsize $\gamma_t \in (0, 1]$, to allow $J_i^{t+1}(x)$ to take a higher value than $V_i^t(x)$.

In particular, for $t \in \mathbf{T}_i$, in place of Eq. (4.4), for all $x \in X_i$, we calculate

$$\tilde{J}_i^{t+1}(x) = \left(T_{i,\mu_i^t} \big(J_1^{\tau_{i1}(t)}, \ldots, J_n^{\tau_{in}(t)} \big) \right)(x), \qquad (5.2)$$

and we use the update

$$J_i^{t+1}(x) = \tilde{J}_i^{t+1}(x), \qquad \text{if } \tilde{J}_i^{t+1}(x) \leq V_i^t(x), \qquad (5.3)$$

and

$$J_i^{t+1}(x) = \gamma_t \tilde{J}_i^{t+1}(x) + (1 - \gamma_t) V_i^t(x)$$
$$\text{if } \tilde{J}_i^{t+1}(x) > V_i^t(x), \qquad (5.4)$$

where $\{\gamma_t\}$ is a sequence with $\gamma_t \in (0, 1]$ and $\gamma_t \to 0$. For $t \notin \overline{\mathbf{T}}_i \cup \mathbf{T}_i$, we leave $J_i(x)$ unchanged:

$$J_i^{t+1}(x) = J_i^t(x), \qquad \forall \, t \notin \overline{\mathbf{T}}_i \cup \mathbf{T}_i. \qquad (5.5)$$

As in the algorithm of Section IV, we assume that the initial V^0 satisfies $\underline{J} \leq V^0 \leq \overline{J}$ (for example $V^0 = J^0$), while the initial μ^0 is arbitrary.

The idea of the algorithm is to aim for larger increases of $J_i^t(x)$ when the condition (5.1) holds. Asymptotically, as $\gamma_t \to 0$, the iteration (5.2)-(5.4) becomes identical to the convergent update (4.4). It can be shown that if the condition $J^0 \geq T_{\mu^0} J^0$ is satisfied, and $\tau_{ij}(t)$ are monotonically increasing with t, then the convergence of iteration (5.2)-(5.4) is monotonic from above to J^* and we have

$$\tilde{J}_i^{t+1}(x) \leq V_i^t(x)$$

at all times, so that from Eq. (5.3), $J_i^{t+1}(x) = \tilde{J}_i^{t+1}(x)$. The same is true for the algorithm of Section IV, i.e., the
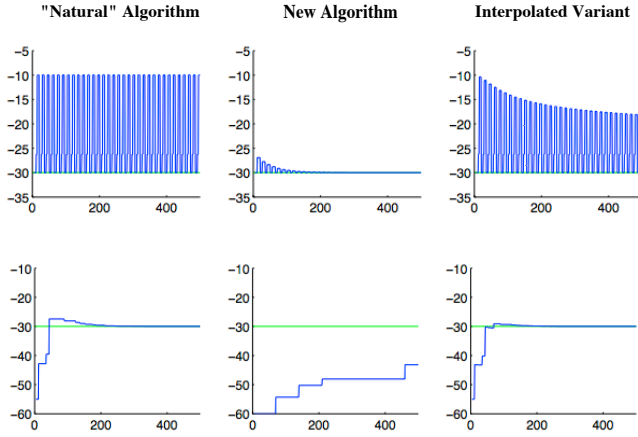
Fig. 5.1. Comparison of three algorithms for a 6-state Williams-Baird counterexample. Plotted are $J^t(x)$ for $x = 2$ under a malicious order of component selection with a malicious choice of J^0 (top row), and under a random order of component selection with J^0 well below J^* (bottom row).

minimum over $V_i^t(x)$ in Eq. (4.4) is unnecessary. Thus in the case where $J^0 \geq T_{\mu^0} J^0$, the algorithms of Sections IV and V coincide with the "natural" asynchronous policy iteration algorithm, and the convergence result of [WiB93] mentioned in the introduction is recovered.

Figure 5.1 illustrates the behavior of these three algorithms for Example 2 of Williams and Baird [WiB93] (with $\alpha = 0.9$). The top row shows how our algorithms correct the convergence difficulty of the natural asynchronous policy iteration algorithm (1.4)-(1.5), while the bottom row shows how the algorithm of this section can improve over the algorithm of Section IV when J^t approaches J^* from below.

The convergence properties of the algorithm with the modification (5.2)-(5.4) will be more fully discussed elsewhere. The following proposition proves convergence under some additional assumptions, which are satisfied in the discounted finite-state MDP Example 2.1, and in SSP models where all policies are proper (see the proof of Prop. 2.2.3 of [Ber07]).

Proposition 5.1: *Let Assumptions 3.1-3.3 hold. Assume further that $\underline{J}$ and $\overline{J}$ are real-valued and bounded, and that for all J with $\underline{J} \leq J \leq \overline{J}$, we have*

$$T_\mu J \leq \overline{J}, \quad \forall \ \mu \in \mathcal{M}, \qquad (5.6)$$

and

$$T(J + c\mathbf{1}) \leq TJ + c\mathbf{1}, \quad \forall \ c \geq 0, \qquad (5.7)$$

where $\mathbf{1}$ denotes the function that is identically equal to 1. Then for all $i = 1, \ldots, n$, and $x \in X_i$, we have

$$\lim_{t \to \infty} J_i^t(x) = \lim_{t \to \infty} V_i^t(x) = J^*(x).$$

Proof: For any $\delta > 0, k > 0$, we will show that for all $i = 1, \ldots, n$, and $x \in X_i$,

$$(T^k \underline{J})(x) \leq J_i^t(x) \leq (T^k \overline{J})(x) + \delta,$$

$$(T^k \underline{J})(x) \leq V_i^t(x) \leq (T^k \overline{J})(x) + \delta,$$

for all t sufficiently large, which will imply the desired conclusion in view of Assumption 3.1(c). Since by using

the update formula (5.3)-(5.4) one can only increase J_i^t for $t \in \mathbf{T}_i$ than by using the update formula (4.4) of the basic algorithm, the lower bounds in the above inequalities follow from the same monotonicity arguments as in the proof of Prop. 4.1, and we only need to prove the upperbounds.

Let $\Delta = \|\overline{J} - \underline{J}\|_\infty$, which is finite because $\underline{J}$ and $\overline{J}$ are bounded under our assumption. Let $\bar{\gamma}_k, k \geq 1$, be positive scalars such that $\Delta \sum_{k=1}^\infty \bar{\gamma}_k \leq \delta$. Define $\delta_k = \Delta \sum_{\ell=1}^k \bar{\gamma}_\ell$ for $k \geq 1$ and $\delta_0 = 0$.

We will show that there exist integers $t_k, k \geq 0$, such that

$$J_i^t(x) \leq (T^k \overline{J})(x) + \delta_k, \quad \forall \ x \in X_i, \text{ and } t \geq t_k, \quad (5.8)$$

$$V_i^t(x) \leq (T^k \overline{J})(x) + \delta_k, \quad \forall \ x \in X_i, \text{ and } t \geq t_k, \quad (5.9)$$

and for all $i, j = 1, \ldots, n$,

$$J_i^{\tau_{ji}(t)}(x) \leq (T^k \overline{J})(x) + \delta_k, \quad \forall \ x \in X_i, \ t \geq t_k. \quad (5.10)$$

The proof is by induction on k.

For $k = 0$, with $\delta_0 = 0$ and $t_0 = 0$, Eqs. (5.8)-(5.10) follow from the initial condition on J^0 and V^0, and Assumptions 3.1 and Eq. (5.6), by induction on t, similar to the proof of Prop. 4.1. In particular, we only need to verify that the update formula (5.4) for J_i^t maintains the relation $J_i^t(x) \leq \overline{J}(x)$. Indeed, if $\bar{t} \in \mathbf{T}_i$ and Eqs. (5.8)-(5.10) hold for all $t \leq \bar{t}$, then from Eqs. (5.10) and (5.2), and Assumptions 3.1 and (5.6), we have

$$\tilde{J}_i^{\bar{t}+1}(x) \leq (T_{i,\mu_i^{\bar{t}}} \overline{J})(x) \leq \overline{J}(x),$$

so if Eq. (5.4) is used to obtain $J_i^{\bar{t}+1}(x)$, then $J_i^{\bar{t}+1}(x) \leq \overline{J}(x)$ in view of Eq. (5.9) and the fact $\gamma_{\bar{t}} \in (0, 1]$. The preceding induction argument also shows that for all $t \in \mathbf{T}_i$,

$$\tilde{J}_i^{t+1}(x) \leq \overline{J}(x) \quad \forall \ i = 1, \ldots, n, \ x \in X_i. \qquad (5.11)$$

Suppose that Eqs. (5.8)-(5.10) hold for some k. We will show that they hold for $k + 1$. For every $i = 1, \ldots, n$, let $\bar{t}_k(i)$ be the first integer $t \geq t_k$ such that $t \in \overline{\mathbf{T}}_i$ (which is finite under Assumption 3.3). Then, using the update formula (4.3) for V^t, Eq. (5.10), and Assumptions 3.1 and (5.7), we have

$$V_i^t(x) \leq (T^{k+1} \overline{J})(x) + \delta_k, \quad \forall \ x \in X_i, \ t > \bar{t}_k(i). \quad (5.12)$$

Consider now J_i^{t+1} for any $t > \bar{t}_k(i)$. Using the update formulas for J_i^t, we see that if $t \in \overline{\mathbf{T}}_i$, or if $t \in \mathbf{T}_i$ and J_i^{t+1} is obtained through the update formula (5.3), we have

$$J_i^{t+1}(x) \leq V_i^t(x) \leq (T^{k+1} \overline{J})(x) + \delta_k, \quad \forall \ x \in X_i,$$

while if $t \in \mathbf{T}_i$ and the update formula (5.4) is used to obtain J_i^{t+1}, by using Eqs. (5.12) and (5.11), we have for $x \in X_i$,

$$\begin{aligned} J_i^{t+1}(x) &= (1 - \gamma_t) V_i^t(x) + \gamma_t \tilde{J}_i^{t+1}(x) \\ &\leq (1 - \gamma_t)\big((T^{k+1} \overline{J})(x) + \delta_k\big) + \gamma_t \overline{J}(x) \\ &\leq (T^{k+1} \overline{J})(x) + \delta_k + \gamma_t\big(\overline{J}(x) - (T^{k+1} \overline{J})(x)\big) \\ &\leq (T^{k+1} \overline{J})(x) + \delta_k + \gamma_t \Delta, \end{aligned}$$

where the last inequality follows from Assumption 3.1 and the definition of Δ. Thus, letting $\tilde{t}_{k+1}$ be such that

$$\tilde{t}_{k+1} > \max_{i=1,\ldots,n} \bar{t}_k(i) \quad \text{and} \quad \gamma_t \leq \bar{\gamma}_{k+1}, \quad \forall \ t \geq \tilde{t}_{k+1},$$

we have for all $i = 1, \ldots, n$,

$$J_i^{t+1}(x) \le (T^{k+1}\overline{J})(x) + \delta_k + \bar{\gamma}_{k+1}\Delta$$
$$= (T^{k+1}\overline{J})(x) + \delta_{k+1}, \qquad (5.13)$$
$$\forall \, x \in X_i, \ t \ge \tilde{t}_{k+1}.$$

Finally, letting t_{k+1} be an integer such that

$$\tau_{ji}(t) > \tilde{t}_{k+1}, \quad \forall \, t \ge t_{k+1}, \ i,j = 1, \ldots, n,$$

and using Eqs. (5.12) and (5.13), we see that Eqs. (5.8)-(5.10) hold with $k+1$ in place of k. The induction is complete. **Q.E.D.**

As noted earlier, the assumptions (5.6) and (5.7) are satisfied for the discounted finite-state MDP Example 2.1, and for SSP models where all policies are proper. More generally, let TJ and $T_\mu J$ be bounded for all bounded J, and let H satisfy for some $\alpha \in (0,1)$, and all $x \in X$ and $u \in U(x)$ the condition

$$\left| H(x, u, J) - H(x, u, J') \right| \le \alpha \|J - J'\|_\infty, \qquad (5.14)$$

for all bounded J and J'. Then T is a sup-norm contraction, so Eq. (5.7) holds, and all mappings T_μ are sup-norm contractions with corresponding fixed points J_μ. Assume that J_μ are uniformly bounded in the sense that $\sup_{\mu \in \mathcal{M}} \|J_\mu\|_\infty < \infty$ (e.g., when $\mathcal{M}$ is finite). Then any function $\overline{J}$ that satisfies

$$\alpha \|\overline{J} - J_\mu\|_\infty \mathbf{1} \le \overline{J} - J_\mu, \qquad \forall \, \mu \in \mathcal{M}, \qquad (5.15)$$

(for example a sufficiently large constant function) also satisfies the condition (5.6). To see this, note that for $J \le \overline{J}$,

$$T_\mu J \le T_\mu \overline{J} \le J_\mu + \alpha \|\overline{J} - J_\mu\|_\infty \mathbf{1} \le \overline{J},$$

where the first inequality follows from the monotonicity of H, the second inequality follows by applying Eq. (5.14) with $J = \overline{J}$, $J' = J_\mu$, and $u = \mu(x)$, and the third inequality is Eq. (5.15). Note that the condition (5.14) is satisfied for the discounted cost Examples 2.1, 2.3, 2.4, and 2.5.

VI. CONCLUDING REMARKS

We have considered general DP-type problems, and proposed modified policy iteration methods that converge under less restrictive conditions than existing methods. Our methods also apply to broader classes of problems, in both a nondistributed and a totally asynchronous distributed setting. In the latter case, they rectify the convergence difficulties of asynchronous policy iteration demonstrated by Williams and Baird [WiB93]. Future works will provide a more complete discussion of synchronous and asynchronous modified policy iteration algorithms for general DP-like models.

REFERENCES

[Bau78] Baudet, G. M., 1978. "Asynchronous Iterative Methods for Multiprocessors," *J. of the ACM*, Vol. 25, pp. 226-244.

[BeC91] Bertsekas, D. P., and Castañon, D. A., 1991. "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," *Parallel Computing*, Vol. 17, pp. 707-732.

[BeG97] Beraldi, P., and Guerriero, F., 1997. "A Parallel Asynchronous Implementation of the Epsilon-Relaxation Method for the Linear Minimum Cost Flow Problem," *Parallel Computing*, Vol. 23, pp. 1021-1044.

[BeE87] Bertsekas, D. P., and El Baz, D., 1987. "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems," *SIAM J. on Control and Optimization*, Vol. 25, pp. 74-85.

[BeE88] Bertsekas, D. P., and Eckstein, J., 1988. "Dual Coordinate Step Methods for Linear Network Flow Problems," *Math. Programming, Series B*, Vol. 42, pp. 203-243.

[BeS78] Bertsekas, D. P., and Shreve, S. E., 1978. *Stochastic Optimal Control: The Discrete Time Case*, Academic Press, N. Y.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., 1989. *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, N. J; republished by Athena Scientific, Belmont, MA, 1997.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA.

[BeY10] Bertsekas, D. P., and Yu, H., 2010. "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Lab. for Information and Decision Systems Report LIDS-P-2831, Massachusetts Institute of Technology.

[Ber77] Bertsekas, D. P., 1977. "Monotone Mappings with Application in Dynamic Programming," *SIAM J. on Control and Optimization*, Vol. 15, pp. 438-464.

[Ber82] Bertsekas, D. P., 1982. "Distributed Dynamic Programming," *IEEE Transactions on Aut. Control*, Vol. AC-27, pp. 610-616.

[Ber83] Bertsekas, D. P., 1983. "Distributed Asynchronous Computation of Fixed Points," *Mathematical Programming*, Vol. 27, pp. 107-120.

[Ber88] Bertsekas, D. P., 1988. "The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem," *Annals of Operations Research*, Vol. 14, pp. 105-123.

[Ber07] Bertsekas, D. P., 2007. *Dynamic Programming and Optimal Control*, 3rd Edition, Vol. II, Athena Scientific, Belmont, MA.

[Bor98] Borkar, V. S., 1998. "Asynchronous Stochastic Approximation," *SIAM J. on Control and Optimization*, Vol. 36, pp. 840851. (Correction note in ibid., Vol. 38, pp. 662663.)

[ChM69] Chazan and Miranker, 1969. "Chaotic Relaxation," *Lin. Algebra and its Appl.*, Vol. 2, pp. 199-222.

[ESM96] El Baz, D., Spiteri, P., Miellou, J. C., and Gazen, D., 1996. "Asynchronous Iterative Algorithms with Flexible Communication for Nonlinear Network Flow Problems," *J. of Parallel and Distributed Computing*, Vol. 38, pp. 1-15.

[ElT82] El Tarazi, M. N., 1982. "Some Convergence Results for Asynchronous Algorithms," *Numer. Math.*, Vol. 39, pp. 325-340.

[Mie75] Miellou, J. C., 1975. "Algorithmes de Relaxation Chaotique a Retards," *R.A.I.R.O.*, Vol. 9, pp. 55-82.

[Rob76] Robert, F., 1976. "Contraction en Norme Vectorielle: Convergence d' Iterations Chaotiques pour des Equations Non Lineaires de Point Fixe a Plusieurs Variables," *Lin. Algebra and its Appl.*, Vol. 13, pp. 19-35.

[Put94] Puterman, M. L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, J. Wiley, N. Y.

[Spi86] Spiteri, P., 1986. "Parallel Asynchronous Algorithms for Solving Two-Point Boundary Value Problems," in M. Cosnard et. al. (eds), *Parallel and Distributed Architectures*, Elsevier, N. Y., pp. 73-84.

[SuB98] Sutton, R. S., and Barto, A. G., 1998. Reinforcement Learning, MIT Press, Cambridge, MA.

[TBT90] Tseng, P., Bertsekas, D. P., and Tsitsiklis, J. N., 1990. "Partially Asynchronous Parallel Algorithms for Network Flow and Other Problems," *SIAM J. on Control and Optimization*, Vol. 28, pp. 678-710.

[TsV99] Tsitsiklis, J. N., and Van Roy, B., 1999. "Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing Financial Derivatives," *IEEE Transactions on Automatic Control*, Vol. 44, pp. 1840-1851.

[Tsi87] Tsitsiklis, J. N., 1987. "On the Stability of Asynchronous Iterative Processes," *Math. Systems Theory*, Vol. 20, pp. 137-153.

[Tsi94] Tsitsiklis, J. N., 1994. "Asynchronous Stochastic Approximation and Q-Learning," *Machine Learning*, Vol. 16, pp. 185-202.

[WiB93] Williams, R. J., and Baird, L. C., 1993. "Analysis of Some Incremental Variants of Policy Iteration: First Steps Toward Understanding Actor-Critic Learning Systems," Report NU-CCS-93-11, College of Computer Science, Northeastern University, Boston, MA.