PERGAMON

Automatica 38 (2002) 929–943

automatica

www.elsevier.com/locate/automatica

A time aggregation approach to Markov decision processes[☆]

Xi-Ren Cao[a,*,1], Zhiyuan Ren[a,1], Shalabh Bhatnagar[b,2], Michael Fu[b,2], Steven Marcus[b,2]

[a] *Department of Electrical and Electronic Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*

[b] *Institute for Systems Research, University of Maryland at College Park, USA*

Received 6 May 1999; received in revised form 11 October 2001; accepted 19 November 2001

Abstract

We propose a time aggregation approach for the solution of infinite horizon average cost Markov decision processes via policy iteration. In this approach, policy update is only carried out when the process visits a subset of the state space. As in state aggregation, this approach leads to a reduced state space, which may lead to a substantial reduction in computational and storage requirements, especially for problems with certain structural properties. However, in contrast to state aggregation, which generally results in an approximate model due to the loss of Markov property, time aggregation suffers no loss of accuracy, because the Markov property is preserved. Single sample path-based estimation algorithms are developed that allow the time aggregation approach to be implemented on-line for practical systems. Some numerical and simulation examples are presented to illustrate the ideas and potential computational savings. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Time aggregation; Policy iteration; Markov decision processes; Performance potentials; On-line optimization

1. Introduction

Various approximation approaches to reduce the computation in Markov decision processes (MDP) have been proposed. These include approximate policy iteration, aggregation, supervisory control and randomization (Aldhaheri & Khalil, 1991; Bertsekas & Tsitsiklis, 1996; Forestier & Varaiya, 1978; Parr, 1998; Rust, 1997; Sutton, Precup, & Singh, 1999). Approximate policy iteration can be carried out in a number of ways that involve suitable approximation techniques in the evaluation and improvement steps of policy iteration. This approach is called neuro-dynamic programming (Bertsekas & Tsitsiklis, 1996). Examples include Tsitsiklis and Van Roy (1999) and Van Roy, Bertsekas, Lee, and Tsitsiklis (1996). The main approach to aggregation has been to reduce the state space by aggregating the original state space into a more manageable set. However, since state aggregation on a Markov chain fails to preserve the Markov property, exact results based on state aggregation usually require severely restrictive structural properties, for example, see Aldhaheri and Khalil (1991). Thus, this approach is usually implemented as a heuristic approximation.

In this paper, we propose a time aggregation approach for policy iteration of MDPs. Our approach is motivated by Zhang and Ho (1991), where time aggregation was applied to performance sensitivity estimation. Our goal is to reduce computation when the system possesses certain special features, and to propose single sample path-based algorithms for time aggregation to be implemented on-line for real engineering systems. We consider infinite horizon average cost MDP problems. The main idea is as follows. Consider any subset of the state space $\mathscr{S}$, denoted as $\mathscr{S}_1 \subset \mathscr{S}$. Every time the Markov chain reaches a state in $\mathscr{S}_1$, we record the state. The resulting sequence forms an embedded Markov chain, and hence the aggregation is in fact not an approximation. Properly defining the performance function for this embedded chain by aggregating the performance of the original chain on the segment between two consecutive embedded points, we can convert the problem of policy iteration of the original Markov chain on the subset $\mathscr{S}_1$ into the same problem for the embedded Markov chain, which has a smaller state space. For problems in which only a small number of

☆ This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Kenko Uchida under the direction of Editor Tamer Basar.

* Corresponding author. Tel.: +852 2358 7048; Fax: +852 2358 1485.

E-mail addresses: eecao@ee.ust.hk (X.-R. Cao), marcus@cng.umd.edu (S. Marcus).

[1] Supported in part by a grant from Hong Kong RGC (HKUST9203/98E).

[2] Supported in part by the NSF under Grant DMI-9713720, and by the Semiconductor Research Corporations under Grant 97-FJ-491.

states (in $\mathscr{S}_1$) are controllable (i.e., a large number of states for which only a single action is available), the time aggregation approach turns the original problem with a large state space (with $|\mathscr{S}|$ states, where $|A|$ denotes the set cardinality, the number of states in set A) into a problem with a smaller state space (with $|\mathscr{S}_1|$ states). When $|\mathscr{S}_1| \ll |\mathscr{S}|$, the computation reduction can be significant (see the discussion after Algorithm 1 in Section 4 and Example 1 in Section 7).

For problems in which multiple actions are available at almost every state, we may partition the state space into $\mathscr{S} = \mathscr{S}_1 \cup \cdots \cup \mathscr{S}_N$, and apply time aggregation to each subspace one by one sequentially. Thus, the original MDP is a problem with $|\mathscr{S}|$ states, whereas in the time aggregation approach, there are N problems of size $|\mathscr{S}_1|, \ldots, |\mathscr{S}_N|$. It is not clear in this case whether our approach reduces computation (see the discussion after Algorithm 2 in Section 5). However, our approach offers some flexibility in carrying out policy iteration even when all states are controllable. In cases where we know which states are more important than others, our techniques can help in obtaining a "good" policy using less computation, by aggregating states according to "degrees of importance". Thus (for instance), our algorithm can be used to update policies only on the more important states and avoid (altogether) wasting computation on the relatively less important ones (see the discussion in Section 5). Our approach is based on a fundamental concept in MDP, the performance potentials (Xi-Ren Cao, 1999) (or differential cost functions (Gallager, 1996)). The paper consists of two parts; the first part deals with analytical solution and the second develops single sample path-based implementation techniques. In the first part, we find that a straightforward application of time aggregation requires one to estimate the performance potentials for every policy and hence is not practically feasible. We introduce an equivalent problem which has the same optimal policy as the time-aggregated problem for the embedded Markov chain; the equivalent problem can be solved by comparing actions, rather than policies. A policy iteration algorithm based on analytic solutions is presented.

The work of Forestier and Varaiya (1978) is related to a part of our approach. They proposed a hierarchical control structure for large Markov chains, in which the higher level operates at a slower (aggregated) time scale than the lower level. In their model, control at the higher ("supervisor") level requires information accumulated at the lower level; this corresponds to the concept of time aggregation. However, their approach requires evaluation of every policy at each step, and therefore, their approach essentially is not based on policy iteration (the fundamental advantage of traditional policy iteration is lost). In addition, for future research they proposed to directly estimate the parameters including all the transition probabilities (which, among other, requires a considerable amount of memory storage), while in our approach, the policy iteration is directly based on potentials, so in most cases estimation of system parameters is not needed. This point will become clearer later.

In the second part, we propose an algorithm that implements the time aggregation approach on a single sample path. The algorithm is based on importance sampling and performance potentials. We prove that as the length of the sample path goes to infinity, the single sample path-based time aggregation algorithm terminates in an optimal policy. The sample path-based implementation allows the approach to be applied to real systems by observing the system operation history (Xi-Ren Cao & Wan, 1998; Marbach & Tsitsiklis, 2001; Watkins & Dayan, 1992).

The basic principle of importance sampling is that the information about all actions at any state can be extracted from the current sample path. In our single sample path-based approach, this generally requires that for any $i \in \mathscr{S}_1$ and $j \in \mathscr{S}$ the ratio of the transition probabilities under two actions, $p^{\alpha'}(i,j)/p^{\alpha}(i,j)$, is finite and known, where α is the action taken at state i in the current sample path and α' is any other action that is available at i. Two important aspects regarding this requirement are as follows.

(i) This implies that if $p^{\alpha}(i,j) = 0$, then $p^{\alpha'}(i,j) = 0$ for all other α' (see Assumption 1 and the weaker version Assumption 1′ in Section 6). However, in many systems the decisions are on–off type, e.g., α may represent accepting an incoming packet and α', rejecting the packet. In such cases this assumption is violated because $p^{\alpha}(i,i+1) = 1$, $p^{\alpha}(i,i) = 0$ while $p^{\alpha'}(i,i) = 1$, $p^{\alpha'}(i,i+1) = 0$, where i is the number of packets in the system. One easy way to overcome this difficulty is to use randomized policies. For example, β represents a policy that accepts the packet with probability $\sigma > 0$ and rejects the packet with probability $1-\sigma$. Using β as the current policy satisfies the requirement. Some other possible approaches for overcoming this difficulty are discussed in Section 8.

(ii) As the example in Section 2 illustrates, this requirement does not mean that the transition probabilities have to be completely known. In many cases, the ratio depends only on the actions and not on the system behavior. This point is important since it allows the proposed time aggregation approach to be implemented on a sample path that is observed on-line from the operation of a real system without knowing the actual dynamics of the system.

The rest of the paper is organized as follows. We first give a few examples in Section 2 to illustrate the possible applications of the proposed approach. Then we introduce time aggregation in Section 3. We show that by "aggregating" performance between two embedded points, we can define a performance function for the embedded chain such that its steady-state performance is the same as that of the original Markov chain. In Section 4, we propose a time aggregation-based MDP policy iteration algorithm for the case where control can be exercised only on states in $\mathscr{S}_1$. Because at each state the "aggregated" performance function defined for the embedded chain in Section 3 depends on the actions at other states, the standard policy iteration algorithm cannot be applied directly to the embedded chain with the aggregated function. Therefore, we define a new performance function and prove that the policy iteration algorithm for the embedded Markov chain with this performance is equivalent to that of the original Markov chain (Proposition 3). We show that this approach leads to computational savings when $|\mathscr{S}_1|$ is small. In Section 5, we discuss the general case where multiple actions are allowed in all the states. The state space is partitioned into a set of N subsets $\mathscr{S} = \bigcup_{n=1}^{N} \mathscr{S}_n$. The actions for states in one subset $\mathscr{S}_i$, $i = 1, 2, \ldots, N$, are updated by applying the time aggregation approach with policy actions in $\mathscr{S} - \mathscr{S}_i = \bigcup_{n \neq i} \mathscr{S}_n$ being fixed. We prove that iteratively updating the actions in this manner eventually leads to an optimal policy for the original MDP. This approach offers some flexibility in implementing MDP, which may be helpful in finding a near optimal policy. In Section 6, we develop algorithms for estimating the performance potentials and implementing policy iteration for the aggregated problem based on a single sample path of the Markov chain. In Section 7, we present a few numerical and simulation examples to make a comparison between the standard policy iteration and our time aggregation approach (both analytical and sample path based). Section 8 concludes the paper with a summary and a discussion about future research directions.

2. Illustrative examples

We give a few examples to show when the proposed approach can be used. In the first example, we consider a manufacturing system consisting of two machines and N parts, which are circulating between the two machines, as shown in Fig. 1. Machine 1 (M_1) can perform three operations (1, 2, and 3); their service times are exponentially distributed with rates λ_1, λ_2, and λ_3, respectively. Some parts need to go through all three operations in the sequence of 1,2,3; others only require operations 2 and 3; still others only need operation 3. The probabilities that a part belongs to these three types are p_1, p_2, and p_3, respectively. Machine 1 works on only one part at a time. Machine 2 (M_2) has only one operation; its service time is exponential with rate λ_4.

The system can be represented by a special closed queueing network shown in Fig. 1, which can be modeled as a Markov process with states denoted as (n, i), where n is the number of parts at machine 1 and $i = 1, 2, 3$ denotes the operation that machine 1 is performing. A part after completion of service at machine 1 goes to machine 2 with probability $p^{\alpha}(n)$ ($p^{a}(n) \in [0, 1]$, n is the number of parts at machine 1 at the service completion time) or immediately returns to machine 1 with probability $1 - p^{\alpha}(n)$, with the superscript "α" representing an action. This is the only point where control can be exercised in this example. The performance to be optimized is the weighted sum of the two machines' throughputs. Some transition probabilities of the Markov

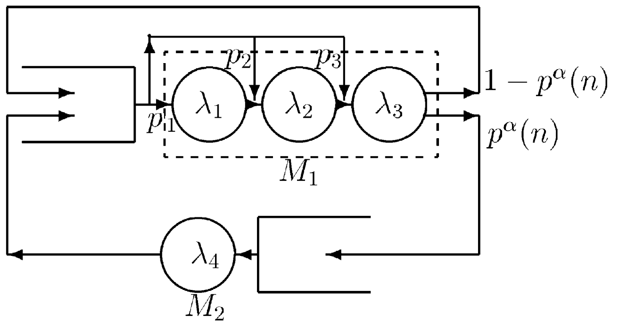

Fig. 1. The queueing network for the manufacturing example.

chain embedded at the operation completion times are (for $0 < n < N$):

$$p[(n,1),(n+1,1)] = \frac{\lambda_4}{\lambda_1 + \lambda_4},$$

$$p[(n,1),(n,2)] = \frac{\lambda_1}{\lambda_1 + \lambda_4},$$

$$p^{\alpha}[(n,3),(n-1,1)] = \frac{\lambda_3}{\lambda_3 + \lambda_4} p_1 p^{\alpha}(n),$$

$$p^{\alpha}[(n,3),(n,1)] = \frac{\lambda_3}{\lambda_3 + \lambda_4} p_1 [1 - p^{\alpha}(n)].$$

Other nonzero transition probabilities have a similar form.

In this example, the state space $\mathscr{S} = \{(n,i)\} \cup \{0\}$, $n = 1, \ldots, N$, $i = 1, 2, 3$ (in general, $i = 1, 2, \ldots, k$, for any integer k) and the controllable states are in $\mathscr{S}_1 = \{(n,3)\}$, $n = 1, \ldots, N$. Therefore, the time aggregation approach can be applied. As shown in Section 6, the sample path-based approach requires knowledge of the ratio of the transition probabilities given a sample path obtained by simulation or observation; in this example, these rates are

$$\frac{p^{\alpha'}[(n,3),(n-1,i)]}{p^{\alpha}[(n,3),(n-1,i)]} = \frac{p^{\alpha'}(n)}{p^{\alpha}(n)}, \quad i = 1, 2, 3$$

and

$$\frac{p^{\alpha'}[(n,3),(n,i)]}{p^{\alpha}[(n,3),(n,i)]} = \frac{1 - p^{\alpha'}(n)}{1 - p^{\alpha}(n)}, \quad i = 1, 2, 3.$$

Thus, the exact transition probabilities are not required for implementing the single sample path-based approach proposed in this paper (see Section 6); rather, the ratio of the transition probabilities depends only on actions, not on the underlying system structure. In other words, one does not need to know λ_i, $i = 1, 2, 3, 4$, and p_j, $j = 1, 2, 3$, as long as a sample path is observed. Many other MDP problems involving routing optimization in queueing systems are of a similar nature.

Another example is a multimedia transmission line carrying two types of packets (e.g., data and video) with different service requirements. While the video packets require a shorter transmission delay, the cost of losing data packets is much higher than the cost of losing video packets. There are two buffers: B_1 for data packets and B_2 for video packets. Each buffer can hold at most N packets. If a data packet arrives and finds that B_1 is full, the packet can be either dropped, leading to a data packet loss, or put in B_2, leading to a longer delay in video packets. If a video packet arrives and finds that B_2 is full, the packet is simply dropped. We assume that the transmission times of each packet (data or video) are exponentially distributed. Defining the state of the system by (n_1, n_2), with n_1 and n_2 representing the number of packets in B_1 and B_2, respectively, a decision on whether or not to drop a data packet is taken only in states (N, n_2) (i.e., the state space of the embedded chain is $\mathscr{S}_1 = \{(N, n_2)\}$, $n_2 = 0, \ldots, N-1$). Applying the time aggregation approach reduces the size of the state space from $(N+1)^2$ to N.

Finally, the proposed approach may be applied to systems with threshold control policies. Denote the state space as $n \in \{0, 1, 2, \ldots\}$ and suppose that the control policy is of the threshold type (e.g., there is an integer B such that if $n \geqslant B$ then action a is taken, and otherwise action b is taken). If a priori we know the bounds of B, (say) $N_1 < B \leqslant N_2$, then we can fix the action for $n > N_2$ and $n \leqslant N_1$ and study the embedded chain for $\mathscr{S}_1 = \{N_1 + 1, \ldots, N_2\}$. This reduces the problem from a state space with infinitely many states to a finite one with $N_2 - N_1$ states.

3. Time aggregation

We introduce the notion of *time* aggregation in this section. Let $\mathrm{X} = \{X_t, t = 0, 1, 2, \ldots\}$ be an ergodic discrete time Markov chain with a finite state space $\mathscr{S} = \{1, 2, \ldots, |\mathscr{S}|\}$ and transition probability matrix $P = [p(i,j)]_{i,j=1}^{|\mathscr{S}|}$. The underlying probability space is denoted as $(\Omega, \mathscr{P}, \Sigma)$; a point $\omega \in \Omega$ corresponds to a sample path of the Markov chain. Let $\pi = [\pi(1), \ldots, \pi(|\mathscr{S}|)]$ be a row vector of the steady-state probabilities, $f = [f(1), \ldots, f(|\mathscr{S}|)]^{\mathrm{T}}$ be a column vector of performance functions. We use $v(i)$ to denote the ith element of a vector v. The finite horizon average cost is

$$\eta_T = \frac{1}{T} \sum_{t=0}^{T-1} f(X_t). \tag{1}$$

By ergodicity, we have for arbitrary initial state

$$\lim_{T \to \infty} \eta_T = \pi f, \quad w.\ p.\ 1.$$

Let $e^{(n)} = (1, 1, \ldots, 1)^{\mathrm{T}}$ be an n-dimensional column vector whose components are all ones (sometimes we simply use e if there is no confusion about the dimension), where the superscript "T" denotes transpose, and I be the identity matrix. Then we have $\pi P = \pi$, $\pi e = 1$, and $Pe = e$. The potential vector $g = [g(1), g(2), \ldots, g(|\mathscr{S}|)]^{\mathrm{T}}$ is defined as the solution to the Poisson equation (Xi-Ren Cao, 1998):

$$(I - P + e\pi) g = f.$$

Now we consider two complementary subsets $\mathscr{S}_1$ and $\mathscr{S}_2 = \mathscr{S} - \mathscr{S}_1$. Without loss of generality, we let $\mathscr{S}_1 = \{1, \ldots, |\mathscr{S}_1|\}$ and $\mathscr{S}_2 = \{|\mathscr{S}_1| + 1, \ldots, |\mathscr{S}|\}$. Consider a sample path $\omega = (X_0, X_1, X_2, \ldots)$ with $X_0 \in \mathscr{S}_1$. Let $t_0 = 0$ and $t_i = \min_t\{t > t_{i-1}, X_t \in \mathscr{S}_1\}$, $i = 1, 2, \ldots$. Then, $\{X_{t_i}, i = 0, 1, 2, \ldots\}$ forms an embedded Markov chain that is also ergodic (Theorem 1 in Forestier and Varaiya (1978), and Revuz (1984, p. 15). Let $\boldsymbol{Y} \triangleq \{Y_i = X_{t_i}, i \geqslant 0\}$, $Y_i \in \mathscr{S}_1$. Let $\tilde{P}$ and $\tilde{\pi}$ denote the transition matrix and steady-state probability row vector of the embedded chain, which satisfy

$$\tilde{\pi} = \tilde{\pi} \tilde{P}. \tag{2}$$

The sample path of the Markov chain is divided into segments, called $\mathscr{S}_1$-segments, by the embedded chain. Denote $\xi_i = (X_{t_i}, X_{t_i+1}, \ldots, X_{t_{i+1}-1})$; ξ_i is called an $\mathscr{S}_1$-segment. We can write $\omega = (\xi_0, \xi_1, \xi_2, \ldots)$, where ξ_i, $i = 1, 2, \ldots$, are random sequences defined on Ω.

Let Ξ be the (countable) set of all feasible $\mathscr{S}_1$-segments. $(\xi_0, \xi_1, \ldots)$ defines a Markov chain with state space Ξ called a *segmented Markov chain*. We use a generic notation $\xi = [\xi(1), \xi(2), \ldots, \xi(n(\xi))]$ to denote an $\mathscr{S}_1$-segment, where $n(\xi)$ is the length of ξ. The infinite segmented chain is irreducible, aperiodic, and positive recurrent, thus ergodic. Its steady-state probability is denoted by $\hat{\pi}(\xi)$. Let E denote the expectation with respect to $\mathscr{P}$ on Ω. Define the quantities

$$h_f(\xi) = \sum_{j=1}^{n(\xi)} f(\xi(j)) \quad \text{and} \quad H_f(i) = E[h_f(\xi) | \xi(1) = i],$$
$$i = 1, 2, \ldots, |\mathscr{S}_1|. \tag{3}$$

Let $H_f = [H_f(1), H_f(2), \ldots, H_f(|\mathscr{S}_1|)]^{\mathrm{T}}$. We will use the notation $h_{\mathbf{1}}$ and $H_{\mathbf{1}}$ above for the case $f(i) = 1, \forall i \in \mathscr{S}$. Thus, $h_{\mathbf{1}}(\xi) = n(\xi)$ and $H_{\mathbf{1}}(i) = E[n(\xi) | \xi(1) = i]$.

From ergodicity, it is clear that $|H_f(i)| < \infty$ for any finite function f. Applying the strong law of large numbers to the segmented chain, we have (cf. Eq. (19) in Forestier and Varaiya, 1978)

$$\eta = \lim_{T \to \infty} \eta_T = \lim_{M \to \infty} \frac{\frac{1}{M} \sum_{m=0}^{M-1} h_f(\xi_m)}{\frac{1}{M} \sum_{m=0}^{M-1} h_{\mathbf{1}}(\xi_m)}$$
$$= \frac{\sum_\xi \hat{\pi}(\xi) h_f(\xi)}{\sum_\xi \hat{\pi}(\xi) n(\xi)} = \frac{\tilde{\pi} H_f}{\tilde{\pi} H_{\mathbf{1}}} = \frac{1}{\bar{n}} \tilde{\pi} H_f, \quad w.\ p.\ 1, \tag{4}$$

where $\bar{n} = \tilde{\pi} H_{\mathbf{1}}$ is the mean length of the $\mathscr{S}_1$-segments.

From (4), if we define a new performance vector for the embedded Markov chain,

$$\tilde{f} = \frac{1}{\bar{n}} H_f, \tag{5}$$

then the embedded chain has the same long-term average performance $\eta = \tilde{\pi} \tilde{f}$ as the original one. The values of H_f and $\bar{n}$ depend on the $\mathscr{S}_1$-segments, and can be calculated by using the following two propositions. First, we partition P and f according to $\mathscr{S}_1$ and $\mathscr{S}_2$ as follows:

$$P = \begin{bmatrix} P_{\mathscr{S}_1\mathscr{S}_1} & P_{\mathscr{S}_1\mathscr{S}_2} \\ P_{\mathscr{S}_2\mathscr{S}_1} & P_{\mathscr{S}_2\mathscr{S}_2} \end{bmatrix} \quad f = \begin{bmatrix} f_{\mathscr{S}_1} \\ f_{\mathscr{S}_2} \end{bmatrix}.$$

Eqs. (6) and (7) in Proposition 1 can also be found in Forestier and Varaiya (1978). Here we provide a straightforward proof.

Proposition 1 (cf. Eqs. (12) and (17) in Forestier and Varaiya (1978)). *We have*

$$\tilde{P} = [P_{\mathscr{S}_1\mathscr{S}_1} \quad P_{\mathscr{S}_1\mathscr{S}_2}] \begin{bmatrix} I \\ (I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} P_{\mathscr{S}_2\mathscr{S}_1} \end{bmatrix}$$

$$= P_{\mathscr{S}_1\mathscr{S}_1} + P_{\mathscr{S}_1\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} P_{\mathscr{S}_2\mathscr{S}_1}, \tag{6}$$

$$H_f = f_{\mathscr{S}_1} + P_{\mathscr{S}_1\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} f_{\mathscr{S}_2}. \tag{7}$$

Proof. The transition probability matrix of the embedded Markov chain is

$$\tilde{P} = P_{\mathscr{S}_1\mathscr{S}_1} + P_{\mathscr{S}_1\mathscr{S}_2} \sum_{k=0}^{\infty} P^k_{\mathscr{S}_2\mathscr{S}_2} P_{\mathscr{S}_2\mathscr{S}_1} = P_{\mathscr{S}_1\mathscr{S}_1} + P_{\mathscr{S}_1\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} P_{\mathscr{S}_2\mathscr{S}_1},$$

which leads to (6). Note that $(I - P_{\mathscr{S}_2\mathscr{S}_2})$ is invertible since P is irreducible (Csenki, 1994).

Let $e_i^{(n)}$ be the ith row of the $n \times n$ identity matrix and

$F_{\mathscr{S}_2} = diag\{f_{\mathscr{S}_2}(1), \ldots, f_{\mathscr{S}_2}(|\mathscr{S}_2|)\}$.

Then,

$$\begin{aligned}
&H_f(i) - f_{\mathscr{S}_1}(i) \\
&= E[h_f(\xi)|\xi(1) = i] - f_{\mathscr{S}_1}(i) \\
&= \sum_{n=1}^{\infty} \sum_{j=1}^{n} E[f(\xi(j))|\xi(1) = i, n(\xi) = n] \\
&\quad \mathscr{P}[n(\xi) = n|\xi(1) = i] - f_{\mathscr{S}_1}(i) \\
&= \sum_{n=1}^{\infty} \sum_{j=2}^{n+1} E[f(\xi(j))|\xi(1) = i, n(\xi) = n+1] \\
&\quad \mathscr{P}[n(\xi) = n+1|\xi(1) = i] \\
&= \sum_{n=1}^{\infty} \sum_{j=1}^{n} e_i^{(|\mathscr{S}_1|)} P_{\mathscr{S}_1\mathscr{S}_2} P^{j-1}_{\mathscr{S}_2\mathscr{S}_2} F_{\mathscr{S}_2} P^{n-j}_{\mathscr{S}_2\mathscr{S}_2} P_{\mathscr{S}_2\mathscr{S}_1} e^{(|\mathscr{S}_1|)} \\
&= \sum_{j=1}^{\infty} \sum_{n=j}^{\infty} e_i^{(|\mathscr{S}_1|)} P_{\mathscr{S}_1\mathscr{S}_2} P^{j-1}_{\mathscr{S}_2\mathscr{S}_2} F_{\mathscr{S}_2} P^{n-j}_{\mathscr{S}_2\mathscr{S}_2} (I - P_{\mathscr{S}_2\mathscr{S}_2}) e^{(|\mathscr{S}_2|)} \\
&= \sum_{j=1}^{\infty} e_i^{(|\mathscr{S}_1|)} P_{\mathscr{S}_1\mathscr{S}_2} P^{j-1}_{\mathscr{S}_2\mathscr{S}_2} f_{\mathscr{S}_2} \\
&= e_i^{(|\mathscr{S}_1|)} P_{\mathscr{S}_1\mathscr{S}_2} (I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} f_{\mathscr{S}_2}.
\end{aligned}$$

This directly leads to (7). □

From (6) and (7), $\tilde{\pi}$ can be obtained by solving (2), and

$$\bar{n} = \tilde{\pi} H_{\mathbf{1}} = \tilde{\pi}[e^{|\mathscr{S}_1|} + P_{\mathscr{S}_1,\mathscr{S}_2}(I - P_{\mathscr{S}_2,\mathscr{S}_2})^{-1} e^{|\mathscr{S}_2|}]$$
$$= 1 + \tilde{\pi} P_{\mathscr{S}_1,\mathscr{S}_2}(I - P_{\mathscr{S}_2,\mathscr{S}_2})^{-1} e^{|\mathscr{S}_2|}. \tag{8}$$

Proposition 2. *We have*

$$\tilde{\pi} = \bar{n} \cdot [\pi(1), \pi(2), \ldots, \pi(|\mathscr{S}_1|)], \tag{9}$$

$$\bar{n} = \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)}. \tag{10}$$

Proof. Write π in block partitioned form $[\pi_{\mathscr{S}_1}, \pi_{\mathscr{S}_2}]$. We can verify that (6) and (9) satisfy

$$\begin{aligned}
\pi_{\mathscr{S}_1}\tilde{P} &= \pi_{\mathscr{S}_1}[P_{\mathscr{S}_1\mathscr{S}_1} + P_{\mathscr{S}_1\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} P_{\mathscr{S}_2\mathscr{S}_1}] \\
&= [\pi_{\mathscr{S}_1} - \pi_{\mathscr{S}_2} P_{\mathscr{S}_2\mathscr{S}_1} + \pi_{\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2}) \\
&\quad \times (I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} P_{\mathscr{S}_2\mathscr{S}_1}] \\
&= \pi_{\mathscr{S}_1}.
\end{aligned}$$

Since the embedded chain is ergodic, the steady-state probability has the unique form given by

$$\begin{aligned}
\tilde{\pi} &= \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)} \cdot [\pi(1), \pi(2), \ldots, \pi(|\mathscr{S}_1|)] \\
&= \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)} \pi_{\mathscr{S}_1}.
\end{aligned}$$

Eq. (10) follows directly from (8) and

$$\begin{aligned}
\bar{n} = \tilde{\pi} H_{\mathbf{1}} &= \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)} \pi_{\mathscr{S}_1} H_{\mathbf{1}} \\
&= \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)} \\
&\quad \times [\pi_{\mathscr{S}_1} e^{(|\mathscr{S}_1|)} + \pi_{\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1} e^{(|\mathscr{S}_2|)}] \\
&= \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)} (\pi_{\mathscr{S}_1} e^{(|\mathscr{S}_1|)} + \pi_{\mathscr{S}_2} e^{(|\mathscr{S}_2|)}) \\
&= \frac{1}{\pi(1) + \pi(2) + \cdots + \pi(|\mathscr{S}_1|)}.
\end{aligned}$$

Therefore, we have (9) as well, thus completing the proof. □

From the above discussion, if we define the performance function for the embedded chain as in (5), then the performance of the embedded chain is the same as the original one. The values of the performance function at the states in $\mathscr{S}_1$ are in fact the average on a corresponding $\mathscr{S}_1$-segment. From a sample path point of view, it looks as if the entire $\mathscr{S}_1$-segment is aggregated onto the embedded point; thus we call this approach "time aggregation". In the next section, we will discuss how this approach can be incorporated into MDP policy iteration.

4. Time-aggregation based policy iteration

Consider a Markov chain $\boldsymbol{X} = \{X_t, t \geqslant 0\}$ with state space $\mathscr{S} = \{1, 2, \ldots, |\mathscr{S}|\}$. At any time $t \geqslant 0$, an action is taken from an action space $\mathscr{A}$ and is applied to $\boldsymbol{X}$. The number of actions is finite, and we consider only stationary policies. Let $\mathscr{A}(i) \subseteq \mathscr{A}$ be the set of actions available in state $i \in \mathscr{S}$. A stationary policy is a mapping $\mathscr{L}: \mathscr{S} \to \mathscr{A}$, i.e., for any state i, $\mathscr{L}$ specifies an action $\mathscr{L}(i) \in \mathscr{A}(i)$. Let $\mathscr{E}$ be the policy space. If action α is taken when the system is in state i, the transition probabilities from state i are denoted by $p^\alpha(i,j)$, $j = 1,2,\ldots,|\mathscr{S}|$. With a policy $\mathscr{L}$, the Markov chain evolves according to the transition matrix $P^{\mathscr{L}} = [p^{\mathscr{L}(i)}(i,j)]_{i,j=1}^{|\mathscr{S}|}$. We assume that the Markov chain is ergodic under all policies and the steady-state probabilities corresponding to the Markov chain running under policy $\mathscr{L}$ are denoted as a row vector $\pi^{\mathscr{L}}$. Let $f^{\mathscr{L}}$ be the column vector of performance functions corresponding to policy $\mathscr{L}$. The steady-state performance is $\eta^{\mathscr{L}} = \pi^{\mathscr{L}} f^{\mathscr{L}}$, and our objective is to minimize $\eta^{\mathscr{L}}$ over the policy space $\mathscr{E}$.

First we assume that actions can only be taken for states in $\mathscr{S}_1$ and the transition probabilities for states in $\mathscr{S}_2$ do not depend on actions. Thus, $P_{\mathscr{S}_2,\mathscr{S}_1}$ and $P_{\mathscr{S}_2,\mathscr{S}_2}$ are fixed. We have

$$P^{\mathscr{L}} = \begin{bmatrix} P^{\mathscr{L}}_{\mathscr{S}_1\mathscr{S}_1} & P^{\mathscr{L}}_{\mathscr{S}_1\mathscr{S}_2} \\ P_{\mathscr{S}_2\mathscr{S}_1} & P_{\mathscr{S}_2\mathscr{S}_2} \end{bmatrix} \quad \text{and} \quad f^{\mathscr{L}} = \begin{bmatrix} f^{\mathscr{L}}_{\mathscr{S}_1} \\ f_{\mathscr{S}_2} \end{bmatrix},$$

i.e., $P_{\mathscr{S}_2\mathscr{S}_1}$, $P_{\mathscr{S}_2\mathscr{S}_2}$, and $f_{\mathscr{S}_2}$ are independent of $\mathscr{L}$.

We use the same notation as in Section 3 except that symbols indicating actions and policies are added when needed, e.g., $\tilde{p}^\alpha(i,\cdot)$ and $H_f(i,\alpha)$ for action α, and $H_f^{\mathscr{L}}$ with $H_f^{\mathscr{L}}(i) = H_f(i,\mathscr{L}(i))$ for policy $\mathscr{L}$. In a standard policy iteration for the embedded Markov chain, at the kth iteration we first solve the Poisson equation

$$(I - \tilde{P}^{\mathscr{L}_k} + e\tilde{\pi}^{\mathscr{L}_k})\tilde{g}^{\mathscr{L}_k} = \tilde{f}^{\mathscr{L}_k}$$

for the potential vector $\tilde{g}^{\mathscr{L}_k}$, where $\mathscr{L}_k$ is the policy at the kth iteration. The policy improvement step is carried out by setting $\mathscr{L}_{k+1} = \arg_{\mathscr{L}\in\mathscr{E}}\{\min[\tilde{P}^{\mathscr{L}}\tilde{g}^{\mathscr{L}_k} + \tilde{f}^{\mathscr{L}}]\}$ as the policy for the $(k+1)$th iteration.

However, there is one major obstacle here: According to (5), the performance function $\tilde{f}^{\mathscr{L}}$ depends on the average length $\bar{n}^{\mathscr{L}}$. In other words, $\tilde{f}^{\mathscr{L}}(i)$ depends on actions taken in states other than i. Therefore, standard policy iteration cannot be implemented on the embedded Markov chain using $\tilde{f}^{\mathscr{L}}(i)$. The solution based on this aggregated performance requires calculating $\bar{n}^{\mathscr{L}}$ for every policy, which is not feasible practically. To overcome this obstacle, in the following, we introduce a new performance function; the policy iteration algorithm based on this performance function reaches the same optimal policy without encountering the aforementioned problem.

Recall that the embedded Markov chain is $\boldsymbol{Y} = \{Y_m = X_{t_m},\ m \geqslant 0\}$, $Y_m \in \mathscr{S}_1$. An action $\alpha \in \mathscr{A}(i)$ determines transition probabilities $\tilde{p}^\alpha(i,j)$, $j = 1,2,\ldots,|\mathscr{S}_1|$. Define a performance function

$$r_\delta(i,\alpha) = H_f(i,\alpha) - \delta H_{\mathbf{1}}(i,\alpha) \qquad (11)$$

at state i of Y, where δ is a real parameter. A policy $\mathscr{L} \in \mathscr{E}$ determines a transition probability matrix $\tilde{P}^{\mathscr{L}} = [\tilde{p}^{\mathscr{L}(i)}(i,j)]_{i,j=1}^{|\mathscr{S}_1|}$ and a performance function vector

$$r_\delta^{\mathscr{L}} = H_f^{\mathscr{L}} - \delta H_{\mathbf{1}}^{\mathscr{L}} \qquad (12)$$

for the Markov chain Y. The steady-state probability row vector is $\tilde{\pi}^{\mathscr{L}}$, and using (4), the steady-state performance is given by

$$\lambda_\delta^{\mathscr{L}} = \tilde{\pi}^{\mathscr{L}} r_\delta^{\mathscr{L}} = \tilde{\pi}^{\mathscr{L}} H_{\mathbf{1}}^{\mathscr{L}}(\eta^{\mathscr{L}} - \delta). \qquad (13)$$

The next proposition shows that using performance function $r_\delta^{\mathscr{L}}$ with $\delta = \eta^{\mathscr{L}}$ we can implement standard policy iteration on $\boldsymbol{Y}$ to obtain the optimal policy for the original problem.

Proposition 3. (i) *If $\mathscr{L}'$ is a policy better than $\mathscr{L}$ for the embedded Markov chain with $\delta = \eta^{\mathscr{L}}$ in performance function* (11), *then $\mathscr{L}'$ is also better than $\mathscr{L}$ for the original Markov chain.*

(ii) *Policy $\mathscr{L}^*$ is optimal for the original Markov chain if and only if $\mathscr{L}^*$ is optimal for the embedded chain with $\delta = \eta^{\mathscr{L}^*}$ in performance function* (11).

Proof. 1. Because $\tilde{\pi}^{\mathscr{L}} H_{\mathbf{1}}^{\mathscr{L}} > 0$, from (13), $\eta^{\mathscr{L}'} < \delta = \eta^{\mathscr{L}}$ if and only if $\lambda_\delta^{\mathscr{L}'} < 0 = \lambda_\delta^{\mathscr{L}}$.

2. Also, $\forall \mathscr{L} \in \mathscr{E}$, $\eta^{\mathscr{L}} \geqslant \delta = \eta^{\mathscr{L}^*}$ if and only if $\lambda_\delta^{\mathscr{L}} \geqslant 0 = \lambda_\delta^{\mathscr{L}^*}$. □

With this proposition, we present the following algorithm for policy iteration on the embedded Markov chain (with performance function (11)), which leads to an optimal policy for the original Markov chain.

Algorithm 1 (time-aggregated policy iteration). (i) Choose an initial policy $\mathscr{L}_0$, set $k := 0$.

(ii) At the kth iteration

(a) Calculate $\eta^{\mathscr{L}_k}$ from (4) by using (2), (6), (7), and (8); set $\delta = \eta^{\mathscr{L}_k}$.

(b) Solve

$$(I - \tilde{P}^{\mathscr{L}_k} + e\tilde{\pi}^{\mathscr{L}_k})\tilde{g}^{\mathscr{L}_k} = r_\delta^{\mathscr{L}_k} = H_f^{\mathscr{L}_k} - \eta^{\mathscr{L}_k} H_{\mathbf{1}}^{\mathscr{L}_k} \qquad (14)$$

for the potential vector $\tilde{g}^{\mathscr{L}_k}$.

(c) Determine the policy for the next iteration $\mathscr{L}_{k+1}$ by minimizing (over all $\mathscr{L} \in \mathscr{E}$)

$$\begin{aligned} &\tilde{P}^{\mathscr{L}}\tilde{g}^{\mathscr{L}_k} + r^{\mathscr{L}}_{\eta^{\mathscr{L}_k}} \\ &= [P^{\mathscr{L}}_{\mathscr{S}_1\mathscr{S}_1} + P^{\mathscr{L}}_{\mathscr{S}_1\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1}P_{\mathscr{S}_2\mathscr{S}_1}]\tilde{g}^{\mathscr{L}_k} \\ &\quad + P^{\mathscr{L}}_{\mathscr{S}_1\mathscr{S}_2}(I - P_{\mathscr{S}_2\mathscr{S}_2})^{-1}[f_{\mathscr{S}_2} - \eta^{\mathscr{L}_k} e^{(|\mathscr{S}_2|)}] \\ &\quad + [f^{\mathscr{L}}_{\mathscr{S}_1} - \eta^{\mathscr{L}_k} e^{(|\mathscr{S}_1|)}]. \end{aligned}$$

```
    In case there are multiple minimizing actions,
    if $\mathscr{L}_k(i)$ is among them, we choose $\mathscr{L}_{k+1}(i) = \mathscr{L}_k(i)$; otherwise, let $\mathscr{L}_{k+1}(i)$ be any one of the
    candidates.
(iii) If $\mathscr{L}_{k+1} = \mathscr{L}_k$, then exit; otherwise set $k := k + 1$
and go to step 2.
```

When there are multiple actions that minimize the quantity in step ii(c), in principle any such action can be chosen for the next step. However, if the current action reaches the minimum, it is retained to avoid possible oscillation between policies.

Proposition 4. *Algorithm* 1 *terminates at an optimal policy for the original Markov chain in a finite number of iterations.*

Proof. The first part of Proposition 3 guarantees performance improvement for the original Markov chain after each iteration. Since the policy space is finite, the algorithm must terminate in a finite number of iterations. The second part of Proposition 3 shows that the final policy must be optimal for the original chain. □

This algorithm saves considerable computation if $|\mathscr{S}_1|$ is small. The major computation involved in the algorithm consists of three parts: (i) Calculating $(I - P_{\mathscr{S}_2,\mathscr{S}_2})^{-1}$ (i.e., the inverse of a $|\mathscr{S}_2| \times |\mathscr{S}_2|$ matrix) once; (ii) Solving $\tilde{\pi}\tilde{P}^{\mathscr{L}_k} = \tilde{\pi}$ for the embedded chain (for $|\mathscr{S}_1|$ unknowns) at each iteration; and (iii) Solving the Poisson equation (14) (again for $|\mathscr{S}_1|$ unknowns) at each iteration. The total computation is roughly of the order $(|\mathscr{S}_2|^3 + 2L_1|\mathscr{S}_1|^3)$, where L_1 denotes the number of iterations required for convergence. On the other hand, the complexity for routine policy iteration is of the order $(2L_2.(|\mathscr{S}|)^3)$, where L_2 is the number of iterations required in finding an optimal policy. If $|\mathscr{S}_1|$ is small, then Algorithm 1 would require a computation of order $|\mathscr{S}_2|^3$ (inverting the matrix $(I - P_{\mathscr{S}_2,\mathscr{S}_2})$ only once), while routine policy iteration would require $2L_2|\mathscr{S}|^3$ (solving the Poisson equation at every iteration).

5. The general case

In the previous section, we assumed that the Markov chain is not controllable on some subset ($\mathscr{S}_2$) of the state space. Now, we consider the general case in which the Markov chain is controllable in every state. We first partition the state space as $\mathscr{S} = \bigcup_{n=1}^{N} \mathscr{S}_n, N \geqslant 1$, with $\mathscr{S}_i \cap \mathscr{S}_j = \emptyset$, $i \neq j$, $i, j = 1, \ldots, N$. It is clear that for any $n \in \{1, 2, \ldots, N\}$, we can fix actions for states in $\mathscr{S} - \mathscr{S}_n$ and apply Algorithm 1 to update actions for states in $\mathscr{S}_n$. Each such update improves the performance. More precisely, we propose the following algorithm for policy iteration in the general case.

```
Algorithm 2 (policy iteration). (i) Choose an initial policy
$\mathscr{L}_0$ and set policy $\mathscr{L} = \mathscr{L}_0$, set $k := 0$, $n := 1$, and $w := 0$.
  (ii) Fix the actions for states in $\mathscr{S} - \mathscr{S}_n$, apply Algorithm
1 to $\mathscr{S}_n$ and $\mathscr{S} - \mathscr{S}_n$ to obtain the best policy (called a
partial optimal policy), denoted by $\mathscr{L}_k$.
  (iii)
      (a) If $\eta^{\mathscr{L}_k} < \eta^{\mathscr{L}_{k-1}}$, then set $k := k + 1$, $n := (n + 1)$
          $\bmod N$, $w := 0$ and go to step 2.
      (b) If $\eta^{\mathscr{L}_k} = \eta^{\mathscr{L}_{k-1}}$, then
            if $w = N$, then exit.
            if $w < N$, then set $w := w + 1$, $n = (n + 1)$
            $\bmod N$, and go to step (ii).
```

In Algorithm 2, $n \bmod N = n$, if $n \leqslant N$, and $(N + 1) \bmod N = 1$; k records the number of partial optimal policies that have been reached (two or more consecutive partial optimal policies may be equal); and w records the number of consecutive partial optimal policies that have the same value. The algorithm terminates when $w = N$.

Proposition 5. *Algorithm* 2 *terminates at a global optimal policy in a finite number of iterations.*

Proof. In step (ii), we apply the time-aggregated policy iteration algorithm (Algorithm 1) to the problem with the actions in $N - 1$ subsets of states fixed. This leads to a partial optimal policy. Since the performance of the next partial optimal policy is always no worse than the last one and there are only a finite number of policies, the algorithm must stop in a finite number of iterations.

Let $\mathscr{L}$ be the final policy obtained by the algorithm and $g^{\mathscr{L}}$ be the potential vector of the Markov chain under $\mathscr{L}$. By the stopping criteria of Algorithm 2, when the algorithm terminates at $\mathscr{L}$, there are already N consecutive partial optimal policies with the same performance value. Since in step (ii)(c) in Algorithm 1 we select the same policy when there is no performance improvement, all the N consecutive partial optimal policies turn out to be the same as $\mathscr{L}$. Therefore,

$$P^{\mathscr{L}} g^{\mathscr{L}} + f^{\mathscr{L}} \leqslant P^{\mathscr{L}'} g^{\mathscr{L}} + f^{\mathscr{L}'}$$

for all $\mathscr{L}' \in \mathscr{E}$, componentwise; that is, $\mathscr{L}$ is a global optimal policy. □

When $N = 1$, Algorithm 2 is the standard policy iteration algorithm. When $N = |\mathscr{S}|$, i.e., all the subsets are singletons, the actions are updated one state at a time. In this case, every state just takes the action that minimizes the performance with the actions of other states fixed. Such a state-by-state procedure thus eventually terminates at a globally optimal policy.

The computational complexity of Algorithm 2 is of the order $\sum_{n=1}^{N} (|\mathscr{S} - \mathscr{S}_n|^3 + 2L_1(n).|\mathscr{S}_n|^3)L(n)$, where $L(n)$ is the number of times one needs to compute partial optimal policies for state groups $\mathscr{S}_n$. The complexity for regular policy iteration in this case is of the order $2L_2(|S_1|+|S_2|+\cdots+|S_N|)^3$.

In general, it is not clear whether Algorithm 2 can offer significant computational savings. However, this algorithm does offer some additional flexibility in implementing policy iteration. For example, suppose that we know a priori that the actions in some states are more important than those in others (for example, these states are visited more often), then we can partition $\mathscr{S}$ into $|\mathscr{S}_1|,\ldots,|\mathscr{S}_N|$ such that actions taken in states in $\mathscr{S}_i$ are more important than those in $\mathscr{S}_{i+1}$. Applying our approach, we can stop policy iteration when we find that the performance reaches an acceptable level, whereas in routine policy iteration, the actions in all the states are updated simultaneously. Moreover, our approach may have the potential of performing better than the routine policy iteration for problems with pre-specified computational budget constraints (e.g., finding a good policy within 10 s). This requires further investigation.

6. Single sample path-based results

In Xi-Ren Cao (1998), it is shown that the potentials can be estimated based on a single sample path of a Markov chain; thus one can implement policy iteration without solving the Poisson equation. In this section, we extend the potential based single sample path approach to the time aggregation problem. The task is more complicated because the effect of action α on $\tilde{p}^{\alpha}(i,j)$ is not explicitly known.

We will concentrate on step (ii) of Algorithm 1. Note that $\sum_{j=1}^{|\mathscr{S}_1|}\tilde{p}^{\alpha}(i,j)\tilde{g}^{\mathscr{L}_k}(j)+r_{\delta}(i,\alpha)$, $\delta=\eta^{\mathscr{L}_k}$ is needed for every $\alpha\in\mathscr{A}(i)$, $i=1,2,\ldots,|\mathscr{S}_1|$, in step (ii)(c) in Algorithm 1. We propose to use importance sampling combined with potential estimation to estimate these quantities along a single sample path of the Markov chain running under policy $\mathscr{L}_k$.

Assumption 1. For all $i\in\mathscr{S}_1$ and $j\in\mathscr{S}$, if there exists $\alpha'\in\mathscr{A}(i)$ such that $p^{\alpha'}(i,j)>0$, then for all $\alpha\in\mathscr{A}(i)$, $p^{\alpha}(i,j)>0$.

This is a standard assumption needed for importance sampling; it assures that the information for action α' is contained in the current sample path with action α at state i. This is the main drawback of the importance sampling approach. However, this assumption can be replaced by a weaker one stated as Assumption 1′ later, which can be satisfied by using randomized policies.

To get $\tilde{g}^{\mathscr{L}_k}$, we need to estimate $r_{\delta}^{\mathscr{L}_k}=H_f^{\mathscr{L}_k}-\delta H_{\mathbf{1}}^{\mathscr{L}_k}$. The estimates of $H_f^{\mathscr{L}_k}(i)$, $H_{\mathbf{1}}^{\mathscr{L}_k}(i)$, $\forall i\in\mathscr{S}_1$ (see below) are based on regenerative segments. Recall that $t_l=\min\{t>t_{l-1},X_t\in\mathscr{S}_1\}$, with $t_0=0$. At state Y_m, we have

$$H_{f,m}^{\mathscr{L}_k}(i)=\begin{cases}\frac{1}{\sum_{l=0}^{m-1}1_i(\xi_l)}\sum_{l=0}^{m-1}\left[1_i(\xi_l)\sum_{t=t_l}^{t_{l+1}-1}f^{\mathscr{L}_k}(X_t)\right]\\ \quad\text{if }\sum_{l=0}^{m-1}1_i(\xi_l)\neq 0,\\ 0\\ \quad\text{otherwise,}\end{cases}$$

$$H_{\mathbf{1},m}^{\mathscr{L}_k}(i)=\begin{cases}\frac{1}{\sum_{l=0}^{m-1}1_i(\xi_l)}\sum_{l=0}^{m-1}[1_i(\xi_l)h_{\mathbf{1}}(\xi_l)]\\ \quad\text{if }\sum_{l=0}^{m-1}1_i(\xi_l)\neq 0,\\ 0\\ \quad\text{otherwise,}\end{cases}$$

$$\eta_m^{\mathscr{L}_k}=\frac{1}{\sum_{l=0}^{m-1}h_{\mathbf{1}}^{\mathscr{L}_k}(\xi_l)}\sum_{l=0}^{m-1}h_f^{\mathscr{L}_k}(\xi_l),\tag{15}$$

$$r_{\delta,m}^{\mathscr{L}_k}(i)=H_{f,m}^{\mathscr{L}_k}(i)-\eta_m^{\mathscr{L}_k}H_{\mathbf{1},m}^{\mathscr{L}_k}(i),\tag{16}$$

where $1_i(\xi)=1$, if $\xi(1)=i$; otherwise $1_i(\xi)=0$. Since the first moment of $h_f^{\mathscr{L}_k(i)}$ and $h_{\mathbf{1}}$ are finite because of the ergodicity of the original chain, from the strong law of large numbers, we have

$$\lim_{m\to\infty}\eta_m^{\mathscr{L}_k}=\eta^{\mathscr{L}_k},\quad\text{w. p. 1}\tag{17}$$

and

$$\lim_{m\to\infty}r_{\delta,m}^{\mathscr{L}_k}(i)=r_{\delta}^{\mathscr{L}_k}(i),\quad\text{w. p. 1.}\tag{18}$$

Let $\tilde{g}^{\mathscr{L}_k}$ be a potential vector given by (14). Then for $\delta=\eta^{\mathscr{L}_k}$ (cf. Xi-Ren Cao, 1999),

$$\begin{aligned}\tilde{g}^{\mathscr{L}_k}(i)-\tilde{g}^{\mathscr{L}_k}(j)&=E\left[\sum_{n=0}^{\tilde{\tau}_i^j-1}[r_{\delta}^{\mathscr{L}_k}(Y_n)-\lambda_{\delta}^{\mathscr{L}_k}]\mid Y_0=i\right]\\&=E\left[\sum_{n=0}^{\tilde{\tau}_i^j-1}r_{\delta}^{\mathscr{L}_k}(Y_n)\mid Y_0=i\right],\end{aligned}\tag{19}$$

where the stopping time $\tilde{\tau}_i^j=\min\{n>0,Y_n=j,Y_0=i\}$ is for the Markov chain $\boldsymbol{Y}$ under policy $\mathscr{L}_k$. Note that $\lambda_{\delta}^{\mathscr{L}_k}=0$ with $\delta=\eta^{\mathscr{L}_k}$. Only the differences between the components of $\tilde{g}^{\mathscr{L}_k}$ are important Xi-Ren Cao (1999). The potential that we use is obtained by fixing j and varying i in (19).

For simplicity, let $Y_0=j$. Define regenerative points: $u_0=0$, and $u_{l+1}=\min\{n:n>u_l,Y_n=j\}$, $l\geqslant 0$. For $i\neq j$, define $v_l(i)$ as follows: Consider the set $\{n:u_{l+1}>n>u_l,Y_n=i\}$. If this set is nonempty, define $v_l(i)$ to be the least element in this set (or the first instant n after u_l and before u_{l+1} at which $Y_n=i$); else, if this set is empty, define $v_l(i) = u_{l+1} - 1$. Also, let $\chi_l(i) = 1$, if $\{n: u_{l+1} > n > u_l, Y_n = i\} \neq \emptyset$; and $\chi_l(i) = 0$, otherwise. By the results shown in Xi-Ren Cao (1999), we have the following estimate of $\tilde{g}^{\mathscr{L}_k}(i)$ at state Y_m:

$$\tilde{g}_m^{\mathscr{L}_k}(i) = \begin{cases} \frac{1}{\sum_{l=0}^{K_m-1}\chi_l(i)} \sum_{l=0}^{K_m-1} \left[\chi_l(i) \sum_{q=v_l(i)}^{u_{l+1}-1} r_{\delta,q}^{\mathscr{L}_k}(Y_q) \right], \\ \quad \text{if } \sum_{l=0}^{K_m-1} \chi_l(i) \neq 0, \\ 0 \\ \quad \text{otherwise,} \end{cases} \tag{20}$$

where $K_m + 1$ is the number of regenerative points obtained upto Y_m, and $r_{\delta,q}^{\mathscr{L}_k}$ is given by (16). It is proved in Proposition 6 that this estimate converges to $\tilde{g}^{\mathscr{L}_k}(i)$ w. p. 1, as $m \to \infty$. For $i \in \mathscr{S}_1$ and $\alpha \in \mathscr{A}(i)$, let

$$c^k(i,\alpha) = \sum_{j=1}^{|\mathscr{S}_1|} \tilde{p}^\alpha(i,j)\tilde{g}^{\mathscr{L}_k}(j) + r_\delta(i,\alpha),$$

which is the expected value of $\tilde{g}^{\mathscr{L}_k}(X_{t_{l+1}}) + h_f^\alpha(\xi_l) - \delta h_{\mathbf{1}}(\xi_l)$, given $X_{t_l} = i$ and the action taken at t_l is α. Recall Ξ is the set of all feasible $\mathscr{S}_1$-segments. Let $\Xi_i = \{i, X_{t_l+1}, \ldots, X_{t_{l+1}-1}\}$ be the subset of Ξ that contains all the $\mathscr{S}_1$-segments starting from $X_{t_l} = i$. Note that the probability of $\{X_{t_l}, X_{t_l+1}, \ldots, X_{t_{l+1}}\}$ given that $X_{t_l} = i$ is

$$p^\alpha(i, X_{t_l+1}) p^{\mathscr{L}_k(X_{t_l+1})}(X_{t_l+1}, X_{t_l+2}) \cdots \\ p^{\mathscr{L}_k(X_{t_{l+1}-1})}(X_{t_{l+1}-1}, X_{t_{l+1}}).$$

Thus, we have

$$\begin{aligned} c^k(i,\alpha) &= \sum_{\xi_l \in \Xi_i, X_{t_{l+1}}} [\tilde{g}^{\mathscr{L}_k}(X_{t_{l+1}}) + h_f^\alpha(\xi_l) - \delta h_{\mathbf{1}}(\xi_l)] \\ &\quad \times [p^\alpha(i, X_{t_l+1}) p^{\mathscr{L}_k(X_{t_l+1})}(X_{t_l+1}, X_{t_l+2}) \cdots \\ &\quad\quad p^{\mathscr{L}_k(X_{t_{l+1}-1})}(X_{t_{l+1}-1}, X_{t_{l+1}})] \\ &= \sum_{\xi_l \in \Xi_i, X_{t_{l+1}}} \left[\frac{p^\alpha(X_{t_l}, X_{t_l+1})}{p^{\mathscr{L}_k(X_{t_l})}(X_{t_l}, X_{t_l+1})} \check{h}^{\alpha,k}(\xi_l, X_{t_{l+1}}) \right] \\ &\quad \times [p^{\mathscr{L}_k(X_{t_l})}(i, X_{t_l+1}) p^{\mathscr{L}_k(X_{t_l+1})}(X_{t_l+1}, X_{t_l+2}) \cdots \\ &\quad\quad p^{\mathscr{L}_k(X_{t_{l+1}-1})}(X_{t_{l+1}-1}, X_{t_{l+1}})] \\ &= E\left[\frac{p^\alpha(X_{t_l}, X_{t_l+1})}{p^{\mathscr{L}_k(X_{t_l})}(X_{t_l}, X_{t_l+1})} \check{h}^{\alpha,k}(\xi_l, X_{t_{l+1}}) \,|\, X_{t_l} = i \right], \end{aligned}$$

where $\check{h}^{\alpha,k}(\xi, x) = h_f^\alpha(\xi) - \delta h_{\mathbf{1}}(\xi) + \tilde{g}^{\mathscr{L}_k}(x)$ and the underlying probability measure of the expectation is for the Markov chain under policy $\mathscr{L}_k$. Note that h_f given by (3) is related to the action taken in state i; therefore, the symbol α is added. The estimate of $c^k(i,\alpha)$ is actually based on the technique of *changing measures*, which is the basic idea in *importance sampling*. Assumption 1 makes it feasible to estimate $c^k(i,\alpha), \forall i \in \mathscr{S}_1, \forall \alpha \in \mathscr{A}(i)$, along the sample path of the Markov chain running under policy $\mathscr{L}_k$. After m regenerative periods, we have the following estimator:

$$c_m^k(i,\alpha) = \frac{1}{\sum_{l=0}^{m-1} 1_i(\xi_l)} \sum_{l=0}^{m-1} \left[1_i(\xi_l) \check{h}_l^{\alpha,k}(\xi_l, X_{t_{l+1}}) \right. \\ \left. \times \frac{p^\alpha(X_{t_l}, X_{t_l+1})}{p^{\mathscr{L}_k(X_{t_l})}(X_{t_l}, X_{t_l+1})} \right], \\ \text{if } \sum_{l=0}^{m-1} 1_i(\xi_l) \neq 0, \quad 0 \text{ otherwise,} \tag{21}$$

where $\check{h}_l^{\alpha,k}(\xi, x) = h_f^\alpha(\xi) - \eta_l^{\mathscr{L}_k} h_{\mathbf{1}}(\xi) + \tilde{g}_l^{\mathscr{L}_k}(x)$, $\tilde{g}_l^{\mathscr{L}_k}$ is given by (20) and $\eta_l^{\mathscr{L}_k}$ is given by (15). Now we can prove convergence of the estimates.

Proposition 6. *For every $i \in \mathscr{S}_1$ and every $\alpha \in \mathscr{A}(i)$,*

$$\lim_{m\to\infty} \tilde{g}_m^{\mathscr{L}_k}(i) = \tilde{g}^{\mathscr{L}_k}(i), \quad w.p.\ 1; \tag{22}$$

$$\lim_{m\to\infty} c_m^k(i,\alpha) = c^k(i,\alpha), \quad w.p.\ 1. \tag{23}$$

Proof. Rewrite (20) as

$$\begin{aligned} \tilde{g}_m^{\mathscr{L}_k}(i) &= \frac{1}{\sum_{l=0}^{K_m-1}\chi_l(i)} \sum_{l=0}^{K_m-1} \left[\chi_l(i) \sum_{q=v_l(i)}^{u_{l+1}-1} r_\delta^{\mathscr{L}_k}(Y_q) \right] \\ &\quad + \frac{1}{\frac{1}{K_m}\sum_{l=0}^{K_m-1}\chi_l(i)} \frac{1}{K_m} \\ &\quad \times \sum_{l=0}^{K_m-1} \left\{ \chi_l(i) \sum_{q=v_l(i)}^{u_{l+1}-1} [r_{\delta,q}^{\mathscr{L}_k}(Y_q) - r_\delta^{\mathscr{L}_k}(Y_q)] \right\}. \end{aligned} \tag{24}$$

Noting that $K_m \to \infty$ w.p. 1 as $m \to \infty$, the first term on the right-hand side of (24) will converge to $g^{\mathscr{L}_k}(i)$ w.p. 1 as $m \to \infty$ (Xi-Ren Cao, 1999). The first fraction in the second term on the right-side of (24), $1/(1/K_m \sum_{l=0}^{K_m-1} \chi_l(i))$, will converge to $1/P(\tilde{\tau}_j^i < \tilde{\tau}_j^j)$ w.p. 1 as $m \to \infty$. Let Δ_m denote the second part of the same:

$$\Delta_m = \frac{1}{K_m} \sum_{l=0}^{K_m-1} \left\{ \chi_l(i) \sum_{q=v_l(i)}^{u_{l+1}-1} [r_{\delta,q}^{\mathscr{L}_k}(Y_q) - r_\delta^{\mathscr{L}_k}(Y_q)] \right\}.$$

Noting that the estimate of $r_\delta^{\mathscr{L}_k}$ is based on the strong law of large numbers, we rearrange the sum in Δ_m as follows:

$$\Delta_m = \sum_{s \in \mathscr{S}_1, s \neq j} \frac{N_s}{K_m} \frac{1}{N_s} \sum_{p \in \phi_s} [r_{\delta,p}^{\mathscr{L}_k}(s) - r_\delta^{\mathscr{L}_k}(s)], \tag{25}$$

where N_s is the number of visits to state s before the $(K_m + 1)$th regenerative point along the sample path, and $\forall s \neq j$, $\phi_s = \{p : Y_p = s \text{ and } \exists l, v_l(i) \leqslant p \leqslant u_{l+1} - 1\}$.

We have $N_s/K_m \to \tilde{\pi}^{\mathscr{L}_k}(s)E[\tilde{\tau}_j^j]$ as $m \to \infty$ w.p. 1, and

$$\lim_{N_s\to\infty} \frac{1}{N_s}\sum_{p\in\phi_s}[r^{\mathscr{L}_k}_{\delta,p}(s) - r^{\mathscr{L}_k}_{\delta}(s)] = 0, \quad w.p.\ 1 \tag{26}$$

follows directly from

$$\limsup_{N_s\to\infty}\left|\frac{1}{N_s}\sum_{p\in\phi_s}[r^{\mathscr{L}_k}_{\delta,p}(s) - r^{\mathscr{L}_k}_{\delta}(s)]\right|$$

$$\leqslant \limsup_{N_s\to\infty}\frac{1}{N_s}\sum_{l=0}^{N_s-1}|r^{\mathscr{L}_k}_{\delta,l}(s) - r^{\mathscr{L}_k}_{\delta}(s)|$$

$$= \lim_{l\to\infty}|r^{\mathscr{L}_k}_{\delta,l}(s) - r^{\mathscr{L}_k}_{\delta}(s)| = 0, \quad w.p.\ 1,$$

the last two equalities following from (18). Since $N_s \to \infty$ w.p. 1 as $m \to \infty$, (22) follows directly from (24)–(26).

Finally, by (22) and (17), we have

$$\lim_{l\to\infty}\check{h}^{\alpha,k}_l(\xi,x) = \check{h}^{\alpha,k}(\xi,x), \quad w.p.\ 1$$

and (23) follows easily from (21). □

With the above estimates, the policy iteration in Algorithm 1 can be implemented on a single sample path.

Algorithm 3 (A single sample path-based implementation of step (ii) in Algorithm 1). (i) Choose an integer M.

(ii) From the sample path of Markov chain running under $\mathscr{L}_k$, estimate $\tilde{g}^{\mathscr{L}_k}_M(i)$ and $c^k_M(i,\alpha)$ for all i and α based on (20) and (21).

(iii) For every $i = 1,2,\ldots,|\mathscr{S}_1|$, let

$$\mathscr{L}_{k+1}(i) = \arg\min_{\alpha\in\mathscr{A}(i)} c^k_M(i,\alpha). \tag{27}$$

In case there are multiple minimizing actions in (27), if $\mathscr{L}_k(i)$ is among them, we choose $\mathscr{L}_{k+1}(i) = \mathscr{L}_k(i)$; otherwise, $\mathscr{L}_{k+1}(i)$ is chosen to be any one of the candidate minimizers.

Because of estimation errors, policy iteration in Algorithm 1 may not terminate at an optimal policy. Suppose a sample path containing M $\mathscr{S}_1$-segments is used for estimation in Algorithm 3, and let $\hat{\mathscr{L}}_M$ be the optimal policy obtained by Algorithm 1. Let $c(i,\alpha) = \sum_{j=1}^{|\mathscr{S}_1|}\tilde{p}^{\alpha}(i,j)g^{\hat{\mathscr{L}}_M}(j) + r_\delta(i,\alpha), \forall i\in\mathscr{S}_1, \forall\alpha\in\mathscr{A}(i)$ for $\delta = \eta^{\hat{\mathscr{L}}_M}$, and let $c_M(i,\alpha)$ be the estimate of $c(i,\alpha)$ at the end of M $\mathscr{S}_1$-segments along the sample path of Markov chain under policy $\hat{\mathscr{L}}_M$ (defined as in (21)). Let $\check{\mathscr{L}}$ be the next improved policy obtained if true values were used. Denote the optimal performance value as η^*. Now we prove that $\hat{\mathscr{L}}_M$ will converge to the optimal policy w.p. 1 as $M \to \infty$.

Proposition 7. $\lim_{M\to\infty}\eta^{\hat{\mathscr{L}}_M} = \eta^*$ *w.p.* 1.

Proof. Since the algorithm terminates at $\hat{\mathscr{L}}_M$, we have

$$c_M(i,\hat{\mathscr{L}}_M(i)) \leqslant c_M(i,\check{\mathscr{L}}(i)) \quad \forall i\in\mathscr{S}_1.$$

Therefore,

$$\begin{aligned}\lambda^{\hat{\mathscr{L}}_M}_\delta - \lambda^{\check{\mathscr{L}}}_\delta &= \sum_{i=1}^{|\mathscr{S}_1|}\tilde{\pi}^{\check{\mathscr{L}}}(i)\left[c(i,\hat{\mathscr{L}}_M(i)) - c(i,\check{\mathscr{L}}(i))\right]\\ &\leqslant \sum_{i=1}^{|\mathscr{S}_1|}\tilde{\pi}^{\check{\mathscr{L}}}(i)\left\{\left[c(i,\hat{\mathscr{L}}_M(i)) - c_M(i,\hat{\mathscr{L}}_M(i))\right]\right.\\ &\quad \left. - \left[c(i,\check{\mathscr{L}}(i)) - c_M(i,\check{\mathscr{L}}(i))\right]\right\}.\end{aligned} \tag{28}$$

Note that by (23), the right-hand side of (28) goes to 0 w.p. 1 as $M \to \infty$. From (28), the fact that $\eta^{\check{\mathscr{L}}} \leqslant \eta^{\hat{\mathscr{L}}_M}$ and $\lambda^{\hat{\mathscr{L}}_M}_\delta = 0$, one obtains

$$0 \geqslant \limsup_{M\to\infty}(\eta^{\check{\mathscr{L}}} - \eta^{\hat{\mathscr{L}}_M}) \geqslant \liminf_{M\to\infty}(\eta^{\check{\mathscr{L}}} - \eta^{\hat{\mathscr{L}}_M})$$

$$= \liminf_{M\to\infty}\frac{1}{\tilde{\pi}^{\check{\mathscr{L}}}H^{\check{\mathscr{L}}}_{\mathbf{1}}}\lambda^{\check{\mathscr{L}}}_\delta \geqslant 0, \quad w.p.\ 1.$$

The proposition now follows directly from the fact that the following two events are equal: $\{\eta^{\hat{\mathscr{L}}_M} = \eta^{\check{\mathscr{L}}}\} = \{\eta^{\hat{\mathscr{L}}_M} = \eta^*\}$. □

From Proposition 7, for any given $\varepsilon > 0$, there always exists an integer $M > 0$ (depending on the particular sample path used) such that for all $n \geqslant M$, $|\eta^{\hat{\mathscr{L}}_n} - \eta^*| < \varepsilon$. Moreover, since there are only a finite number of policies, if we choose ε to be smaller than all the differences between the performance measures of any two policies, the above statement further implies that for all $n \geqslant M$, we have $\eta^{\hat{\mathscr{L}}_n} = \eta^*$. Thus, there exists a sample path-dependent integer M such that if one uses that particular value of M (or any other number greater than it) in Algorithm 3 (for that sample path), one is guaranteed an optimal policy.

In practice, it is not clear as to how such a value of M for individual sample paths can a priori be determined. A weaker statement valid over all sample paths makes more practical sense. Since convergence with probability one implies convergence in probability, Proposition 7 thus indicates that for any $\varepsilon > 0$, $\delta > 0$, there exists an M (large enough) such that for all $n \geqslant M$, $Prob\{|\eta^{\hat{\mathscr{L}}_n} - \eta^*| > \varepsilon\} < \delta$. Again, because there are only a finite number of policies, this further implies that for any $\delta > 0$, for all $n \geqslant M$, $Prob\{\eta^{\hat{\mathscr{L}}_n} \neq \eta^*\} < \delta$. Note that this is only an existential result and does not in any way lay out any specific guidelines on how to select such an M. We observed in our experiments that a large enough value of M (chosen arbitrarily as Algorithm 3 prescribes) results in a policy that is close to an optimal policy, or the optimal policy itself (since as described earlier, there are only a finite number of policies to choose from). A similar result as Proposition 7 can be obtained easily for Algorithm 2.

When the state space for the MDP model is large, storage requirements for the algorithm become an issue. Direct application of the algorithms given in Xi-Ren Cao

(1999) to the original model requires storage of a potential vector of $|\mathscr{S}|$ components. By using the estimator (21) in Algorithm 3, the on-line implementation requires storage of size $\sum_{i=1}^{|\mathscr{S}_1|} |\mathscr{A}(i)| \leqslant \max_i |\mathscr{A}(i)||\mathscr{S}_1|$, and at most $\max_i |\mathscr{A}(i)| \max_i |\mathscr{S}_i|$ for Algorithm 2. Thus, with these algorithms, appropriate partitioning can always be applied to circumvent memory storage problems in solving large-scale MDPs. (The storage discussed here is not the storage required for storing the policy itself, it is the additional storage required in the process of determining the policy. This has significant meaning only when the policies can be stored efficiently, e.g., for threshold-type policies only the thresholds have to be stored.)

We now address the issue of selecting an appropriate partition of the state space. Suppose that the state space is evenly partitioned into N subsets. For $N = 1$, Algorithm 2 is exactly the same as the algorithm given in Xi-Ren Cao (1999). When $N = |\mathscr{S}|$, i.e., $\mathscr{S}_i$, $i = 1, 2, \ldots, N$, are all singletons, only one iteration is needed to obtain the optimal policy in Algorithm 1. This is the case that leads to the greatest storage reduction; however, the $|\mathscr{S}|$ steps in each iteration of Algorithm 2 are obviously infeasible when the system is large. A good partition should thus bring about satisfactory storage reduction at an affordable computational cost.

Finally, Assumption 1 can be further weakened to the following assumption.

Assumption 1′. There exists a policy $\mathscr{L}$ such that if $p^{\alpha}(i,j) > 0$ for some $i \in \mathscr{S}_1$, $j \in \mathscr{S}$ and $\alpha \in \mathscr{A}(i)$, then $p^{\mathscr{L}(i)}(i,j) > 0$.

Note that Assumption 1′ can usually be satisfied by an MDP model with randomized policies. We only need a small change in our on-line results in the last section for this new assumption: all the estimation results based on the sample path of the chain under $\mathscr{L}_k$ are obtained based on the sample path under $\mathscr{L}$ instead, i.e., the system is always running under policy $\mathscr{L}$ during the optimization.

7. Numerical and simulation examples

In this section we give some numerical and simulation examples to illustrate the ideas. The first example shows that when the number of controllable states is small, our analytical time aggregation approach (Algorithm 1) saves computation compared with the standard policy iteration approach. The second example illustrates the sample path-based implementation. The third example shows that in some cases the time aggregation sample path-based approach may save transitions compared with standard implementation, even when all the states are controllable.

Example 1. We consider the multimedia example introduced in Section 2. Two types of packets, data and video, share the same transmission line. Both the data and video packets arrive in Poisson processes with rates λ_d and λ_v, respectively. The service times are assumed to be exponential with rates μ_d and μ_v, respectively (the service rates are guaranteed by, e.g., Intserv or Diffserv technology, Kurose & Ross, 2001). Each type of traffic has a buffer, with capacity N_d and N_v (this capacity includes the packet being transmitted), respectively.

The system state is $[n_1, n_2]$, with n_i being the total number of packets in queue i and server i for $i = 1, 2$ ($i = 1$ for data and $i = 2$ for video). The state space is ordered as $\{[N_d, 0], [N_d, 1], \ldots, [N_d, N_v], [0, 0], [0, 1], \ldots, [0, N_v], \ldots, [N_d - 1, 0], [N_d - 1, 1], \ldots, [N_d - 1, N_v]\}$. If a data packet arrives when the system state is $[N_d, n_2]$, $n_2 = 0, 1, \ldots, N_v - 1$, we have to decide whether to put it into queue 2 or drop it, considering the tradeoff between the loss probability of data packets and waiting time of video packets.

At each controllable state $[N_d, n_2]$, $n_2 = 0, 1, \ldots, N_v - 1$, we have two actions $\{0, 1\}$ available, where action 0 means "rejecting" the packet and action 1 means "accepting" the packet and putting it into the video buffer. The cost in state $[n_1, n_2]$ for $n_1 \neq N_d$ is $f([n_1, n_2]) = k_d n_2$ (reflecting the delay of video packets); the cost in state $[N_d, n_2]$ for $n_2 = 0, 1, \ldots, N_v - 1$, after action α is taken, is $f([N_d, n_2], \alpha) = k_p|\alpha - 1| + k_d n_2$, $\alpha \in \{0, 1\}$ (reflecting packet loss); the cost in state $[N_d, N_v]$ is $f([N_d, N_v]) = k_p + k_d N_v$, where k_p, k_d are assigned weights. We consider the case with a high-rate data traffic and a relatively slow video traffic, when admission control can make a significant difference. Thus, in our numerical example, we choose $\lambda_v = 1$, $\mu_v = \frac{1}{0.9}$ (packets/time unit), $\lambda_d = 10\lambda_v$ $\mu_d = 10\mu_v$, $N_d = N_v = 30$, $k_p = 900$, and $k_d = 1$.

We applied our time aggregation approach (Algorithm 1 with $\mathscr{S}_1 = \{[N_d, 0], [N_d, 1], \ldots, [N_d, N_v]\}$) and the standard policy iteration to the example. Both Algorithm 1 and the standard policy iteration algorithm lead to the same iterations shown in Table 1. Here, we represent a policy as a sequence of 30 Boolean numbers in which the number in the ith place specifies the control action at state $[N_d, i]$, for $i = 0, 1, \ldots, 29$. Table 1 shows that, with more "accept" actions taken, the DLP (data loss probability) drops; VLP (video loss probability) and VWT (video waiting time) increase, and η (long run average cost) decreases, as expected.

We performed the numerical calculation on a 400 MHz machine with 192 Mb RAM by using Matlab v5.3 in the Microsoft Windows 2000 operating system. The traditional policy iteration algorithm takes 633 s to get the results in Table 1 while Algorithm 1 only needs 37 s. The computational savings, as analyzed before, are due to the fact that inversion of 961×961 matrices at each iteration in the routine policy iteration is reduced to inversion of 31×31 matrices at each iteration in the time aggregation-based policy iteration.

With the optimal policy, the overflowed data packets are accepted to the video queue when it is either near empty or near full. The overflowed data packets are rejected only when the video queue is filled to medium capacity. The

Table 1
Policy iterations (Algorithm 1 and traditional policy iteration algorithm)

Iteration #	Policy	DLP	VLP	VWT (time units)	η
0	000000000000000000000000000000	0.0044	0.0044	6.8743	11.7369
1	111111111111110000000001111111	0.0019	0.0075	8.0824	10.9489
2	111111111110000000001111111111	0.0022	0.0076	7.8241	10.9091
3	111111111111000000111111111111	0.0019	0.0088	8.0789	10.8976
4	111111111111000001111111111111	0.0018	0.0093	8.1653	10.8950
5	111111111111000011111111111111	0.0016	0.0099	8.2721	10.8941

fundamental reason for this type of behavior is that the loss of video packets is not so important for the QoS (quality of service) in video applications, and thus video packet loss probability is not counted in our performance criterion. The effect of accepting a data packet into the video queue is small when the video queue is near empty or near full.

Example 2. Here we consider the manufacturing system shown in Fig. 1 to demonstrate our sample path-based results. For simplicity, let $\lambda_1=\lambda_2=\lambda_3=\lambda_4=1$. The state space is ordered as $\{(1,3),\ldots,(N,3),(0,0),(1,1),\ldots,(N,1),(1,2),\ldots,(N,2)\}$. We further assume that the manufacturing procedure requires that $100p^0$ percent of parts ($0 < p^0 < 1$) will definitely go back to machine 1 after leaving it, and $100p^1$ percent will enter machine 2 after completing service at machine 1. The remaining $100(1-p^0-p^1)$ percent can go to either machine 1 or 2, depending on the actions. Therefore, when state is $(i,3)$, $i=1,2,\ldots,N$, there are two control actions $\{0,1\}$ available. When action 0 (or 1) is taken, the remaining parts go to machine 2 (or machine 1). Thus, the transition probabilities associated with these two actions (see Table 1) are $p^0(1)=p^0(2)=p^0(3)=1-p^0$, and $p^1(1)=p^1(2)=p^1(3)=p^1$. The cost (control-action unrelated) at state (n,i) is $f((n,i))=-k_1f_1((n,i))-k_2f_2((n,i))$,

$$\begin{bmatrix}
\frac14-|\delta_1| & \frac14+\delta_1^+ & \frac14+\delta_1^+ & \frac14+\delta_1^+ & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0\\
\frac15+\delta_2^- & \frac15-|\delta_2| & \frac15+\delta_2^+ & \frac15+\delta_2^+ & \frac15+\delta_2^+ & 0 & 0 & 0 & 0 & \cdots & 0 & 0\\
\frac16+\delta_3^- & \frac16+\delta_3^- & \frac16-|\delta_3| & \frac16+\delta_3^+ & \frac16+\delta_3^+ & \frac16+\delta_3^+ & 0 & 0 & 0 & \cdots & 0 & 0\\
\frac17+\delta_4^- & \frac17+\delta_4^- & \frac17+\delta_4^- & \frac17-|\delta_4| & \frac17+\delta_4^+ & \frac17+\delta_4^+ & \frac17+\delta_4^+ & 0 & 0 & \cdots & 0 & 0\\
0 & \frac17+\delta_5^- & \frac17+\delta_5^- & \frac17+\delta_5^- & \frac17-|\delta_5| & \frac17+\delta_5^+ & \frac17+\delta_5^+ & \frac17+\delta_5^+ & 0 & \cdots & 0 & 0\\
 & & & & & & \cdots & & & & & \\
0 & 0 & \cdots & 0 & \frac17+\delta_{22}^- & \frac17+\delta_{22}^- & \frac17+\delta_{22}^- & \frac17-|\delta_{22}| & \frac17+\delta_{22}^+ & \frac17+\delta_{22}^+ & \frac17+\delta_{22}^+ & 0\\
0 & 0 & \cdots & 0 & 0 & \frac17+\delta_{23}^- & \frac17+\delta_{23}^- & \frac17+\delta_{23}^- & \frac17-|\delta_{23}| & \frac17+\delta_{23}^+ & \frac17+\delta_{23}^+ & \frac17+\delta_{23}^+\\
0 & 0 & \cdots & 0 & 0 & 0 & \frac16+\delta_{24}^- & \frac16+\delta_{24}^- & \frac16+\delta_{24}^- & \frac16-|\delta_{24}| & \frac16+\delta_{24}^+ & \frac16+\delta_{24}^+\\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \frac15+\delta_{25}^- & \frac15+\delta_{25}^- & \frac15+\delta_{25}^- & \frac15-|\delta_{25}| & \frac15+\delta_{25}^+\\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \frac14+\delta_{26}^- & \frac14+\delta_{26}^- & \frac14+\delta_{26}^- & \frac14-|\delta_{26}|
\end{bmatrix},$$

where $f_1((n,i))=1$ if $n=0$; 0 otherwise, $f_2((n,i))=1$ if $n=N$; 0 otherwise, and k_1, k_2 are assigned weights (to minimize the weighted machine idle time). In the simulation, we choose $N=3$; $p_1=0.2, p_2=0.2, p_3=0.6$; $p^0(1)=p^0(2)=p^0(3)=1-p^0=0.8$, $p^1(1)=p^1(2)=p^1(3)=p^1=0.2$; and $k_1=0.9, k_2=0.1$.

Both the traditional (Xi-Ren Cao, 1999) and the time-aggregated sample path-based policy iterations yield the same iteration sequence as shown in Table 2 (the number at nth place in the second column is the control action for state $[n,3]$, $n=1,2,3$). However, with the time aggregation approach each iteration only requires 5000 transitions to reach the right decision; while with the standard approach proposed in Xi-Ren Cao (1999) each iteration requires 7000 transitions. Another advantage of the time aggregation approach is that one does not require knowledge of the values of p_1, p_2, and p_3; while the standard approach requires the exact knowledge of these parameters. (This point can be verified by checking the optimality equation for both cases; for the time aggregation case, all the parameters except $p^\alpha(n)$ are cancelled out; while for the standard case, p_i, $i=1,2,3$, remain.)

Example 3. This is an example with all states controllable in which the time aggregation approach saves on the number of transitions. We consider a Markov system in which $\mathscr{S}=\{1,2,\ldots,26\}$; $\mathscr{A}=\{-1,0,1\}$; $\mathscr{A}(1)=\{0,1\}$; $\mathscr{A}(26)=\{-1,0\}$; $\mathscr{A}(i)=\{-1,0,1\}, i=2,3,\ldots,25$; the transition matrix under policy $\mathscr{L}$ is

where $\delta_i=-0.1,0,0.1$, if $\mathscr{L}(i)=-1,0,1$, respectively; $\delta_i^-=-\max\{\frac13,1/(i-1)\}\min\{0,\delta_i\}$, and $\delta_i^+=\max\{\frac13,1/(26-i)\}\max\{0,\delta_i\}$. The cost function is $f(i)=1+\frac{99}{25}(i-1)$. The optimal policy $\mathscr{L}^*$ should be $\{0,-1,\ldots,-1\}$, i.e., $\mathscr{L}^*(1)=0$ and $\mathscr{L}^*(i)=-1$, $i=2,3,\ldots,26$, because the state with a lower index has a lower cost.

First, we set $N = 1$. Applying the original algorithm in Xi-Ren Cao (1999) gives the policies shown in Table 3. Next, we set $N = 13$, $\mathscr{S}_i = \{2i-1, 2i\}$, $i = 1, 2, \ldots, 13$. The results of Algorithm 2 are given in Table 4. We observe that with the single sample path-based implementation, Algorithm 2 reaches the optimal policy after one full iteration, requiring about 6.5 million transitions of the chain, whereas the standard policy iteration algorithm proposed in Xi-Ren Cao (1999) did not converge to the optimal policy (but was close to it) after about 80 million transitions. The reduction in number of transitions (using our on-line algorithm) has an important practical significance: if the above algorithm is applied on-line to a communication system, then it would require about 6.5 million events to reach the optimal policy (and in a fast communication system, this may happen in a few minutes), but standard policy iteration on the other hand would be at least ten times slower.

An intuitive explanation of the reduction in the number of transitions is as follows. Recall that the potential $g(i)$ is estimated using the averages of the sums of the performance functions over a sample period starting from state i to the first entry to a particular state in a Markov chain. Compared with the original chain, the embedded chain has fewer states, thus such a period is shorter and hence the variance in estimating the potentials is smaller. The performance function for the embedded chain can be estimated accurately using the average defined in (16). However, with the time aggregation approach there is an additional variance due to importance sampling; this variance is small (large) when the difference among the sample paths for different actions is small (large). Therefore, the time aggregation sample path-based approach may reduce the number of transitions when such differences are small. In Example 2, we observe that when $p^0 = p^1 < 0.1$ (the sample paths for the two actions are considerably different) both the time aggregation and the standard approaches require about the same number of transitions in each iteration. This is another constraint of the importance sampling technique. Further work is needed to make the above argument rigorous.

This example also shows the flexibility of the time aggregation approach: if a performance of 40 is good enough,

Table 2
Policy iteration for both the traditional and the TA-based iteration algorithms

Iteration #	Policy	η
0	000	−0.89
1	111	−0.91
2	110	−0.93

η: $-1\times$ weighted combined machine throughput

Table 3
Policy iterations for traditional algorithms in Example 3

Iteration #	Policy	Estimated performance
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	50.06
1	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0-1-1-1-1-1-1	32.89
2	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0	32.68
3	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0-1	32.60
	Stopped after 80,000,000 total transitions	

Table 4
Policy iterations for TA-based algorithms in Example 3

Iteration	Step	Policy	Estimated performance
0	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	50.1
	2	0-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	49.9
	3	0-1-1-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	49.5
	4	0-1-1-1-1-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	48.4
	5	0-1-1-1-1-1-1-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	46.7
	6	0-1-1-1-1-1-1-1-1-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	44.6
	7	0-1-1-1-1-1-1-1-1-1-1-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	42.3
	8	0-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0 0 0 0 0 0 0 0 0 0 0	39.8
	9	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0 0 0 0 0 0 0 0 0	37.4
	10	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0 0 0 0 0 0 0	35.0
	11	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0 0 0 0 0	33.8
	12	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0 0 0	33.2
	13	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1 0 0	32.7
	14	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1	32.5
1	1–13	0-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1	32.5
		Terminated by stopping rule after 6,521,704 transitions	

we can stop at step 8; if a performance of 35 is satisfactory, we can stop at step 10. In both cases, only actions at some states are updated; thus, further transition reduction can be achieved at the cost of performance. With the standard approach, actions at all states are updated at the same time when all the potentials are estimated. In addition, Algorithm 2 requires only a storage of six potentials, compared with a storage of 26 for the original algorithm. Thus, for this example, the proposed algorithm results in both computational and storage savings.

Finally, since the transition probabilities are known, the problem can be solved by an analytical method (which takes 0.12 s) which is faster than simulation with 6.5 million transitions (which takes 1.2 s). This example simply demonstrates that for real systems with unknown parameters, sample paths can be observed without simulation, and the time aggregation approach may have some advantages over the standard approach.

8. Conclusions

We proposed a time aggregation approach to solving MDPs, by which policy iteration of a large-scale MDP can be replaced by a series of policy iterations on smaller subsets of the original state space. The main results are obtained by using an equivalent performance function for the embedded Markov chain. Single sample path-based algorithms are presented; these algorithms are based on the estimation of performance potentials.

With the computation-based time aggregation approach, computational and storage savings can be achieved if special structural features can be employed (e.g., when a large number of states are uncontrollable). The approach also provides flexibility in finding a near-optimal solution using less computation if a priori knowledge about the importance of states is available. In addition, the time aggregation sample path-based approach may require less knowledge of system parameters compared with the standard one proposed in Xi-Ren Cao (1999).

As shown in two examples, the sample path-based time aggregation approach may save transitions, which is important for real-time applications. In general, such savings can be achieved when the sample paths for different actions are close to each other. Further study is needed to determine why and when this is the case and how to apply it in real applications.

The proposed sample path-based approach is based on importance sampling, which requires that the information pertaining to other actions at a state is contained in the current sample path. This requires that if $p^{\alpha}(i,j)=0$ for the current action α, then $p^{\alpha'}(i,j)=0$ for any other α'. However, if $p^{\alpha'}(i,j)>0$ and is known, the information about the effect of action α' via state j can also be obtained from the current sample path if it ever visits state j. This indicates that more complicated algorithms can be developed for systems where the assumption in this paper may be violated. This is a topic for further research. Another problem for future research is the extension of this approach to other cases such as discounted or episodic MDPs. To do so we first have to extend the performance potential approach to these cases. In a recent work (Xi-Ren Cao, 2000), a unified framework based on performance potentials is presented for both average cost and discounted MDPs.

Acknowledgements

The authors appreciate the reviewers' comments and suggestions which have significantly improved the presentation of the paper.

References

Aldhaheri, R. W., & Khalil, H. K. (1991). Aggregation of the policy iteration method for nearly completely decomposable Markov chains. *IEEE Transactions on Automatic Control*, *36*, 178–187.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Xi-Ren Cao, & Wan, Y. (1998). Algorithms for sensitivity analysis of Markov systems through potentials and perturbation realization. *IEEE Transactions on Control Systems Technology*, *6*, 482–494.

Xi-Ren Cao (1998). The relation among potentials, perturbation analysis, Markov decision processes, and other topics. *Journal of Discrete Event Dynamic Systems*, *8*, 71–87.

Xi-Ren Cao (1999). Single sample path-based optimization of Markov chains. *Journal of Optimization: Theory and Applications*, *100*(3), 527–548.

Xi-Ren Cao (2000). A unified approach to Markov decision problems and performance sensitivity analysis. *Automatica*, *36*, 771–774.

Csenki, A. (1994). *Dependability for systems with a partitioned state space*. Berlin: Springer.

Forestier, J.-P., & Varaiya, P. (1978). Multilayer control of large Markov chains. *IEEE Transactions on Automatic Control*, *AC-23*, 298–305.

Gallager, R. G. (1996). *Discrete stochastic processes*. Boston, MA: Kluwer Academic Publishers.

Kurose, J. F., & Ross, K. W. (2001). *Computer networking: a top–down approach*. Reading, MA: Addison-Wesley.

Marbach, P., & Tsitsiklis, J. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control*, *46*, 191–209.

Parr, R. E. (1998). *Hierarchical control and learning for Markov decision processes*. Ph.D. dissertation, Department of Computer Science, University of California, Berkeley.

Revuz, D. (1984). *Markov chains*. New York: North-Holland.

Rust, J. (1997). Using randomization to break the curse of dimensionality. *Econometrica*, *65*, 487–516.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *12*, 181–211.

Tsitsiklis, J. N., & Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica*, *35*, 1799–1808.

Van Roy, B., Bertsekas, D. P., Lee, Y., & Tsitsiklis, J. N. (1996). A neuro-dynamic programming approach to retailer inventory management. Preprint.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Zhang, B., & Ho, Y. C. (1991). Performance gradient estimation for very large finite Markov chains. *IEEE Transactions on Automatic Control*, *36*, 1218–1227.

Xi-Ren Cao received the M.S. and Ph.D. degrees from Harvard University, in 1981 and 1984, respectively, where he was a research fellow from 1984 to 1986. He then worked as a principal and consultant engineer/engineering manager at Digital Equipment Corporation, USA, until October 1993. Since then, he is a Professor of the Hong Kong University of Science and Technology (HKUST). He is the director of the Center for Networking at HKUST. He held visiting positions at Harvard University, University of Massachusetts at Amherst, AT& T Labs, University of Maryland at College Park, University of Notre Dame, Shanghai Jiaotong University, Nankei University, Tsinghua University, and University of Science and Technology of China.

Xi-Ren Cao owns two patents in data communications and published two books: "Realization Probabilities—the Dynamics of Queuing Systems", Springer Verlag, 1994, and "Perturbation Analysis of Discrete-Event Dynamic Systems", Kluwer Academic Publishers, 1991 (co-authored with Y.C. Ho). He received the Outstanding Transactions Paper Award from the IEEE Control System Society in 1987 and the Outstanding Publication Award from the Institution of Management Science in 1990. He is a Fellow of IEEE, Associate Editor at Large of IEEE Transactions of Automatic Control, and he is/was Board of Governors of IEEE Control Systems Society, associate editor of a number of international journals and chairman of a few technical committees of international professional societies. His current research areas include discrete event dynamic systems, stochastic processes, optimization techniques, and communication systems, signal processing.

Zhiyuan Ren received the B.E. degree in Automatic Control from University of Science and Technology of China in 1994, and the M.Phil. degree in Electrical and Electronic Engineering from the Hong Kong University of Science and Technology in 1998. He is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at Carnegie Mellon University. His primary research interest is analysis and optimization of stochastic systems.

Michael C. Fu is a Professor in the Robert H.~Smith School of Business, with a joint appointment in the Institute for Systems Research and an affiliate faculty position in the Department of Electrical and Computer Engineering, all at the University of Maryland. He received degrees in Mathematics and EE/CS from MIT, and a Ph.D. in Applied Mathematics from Harvard University. His research interests include simulation and applied probability modeling, particularly with applications towards manufacturing systems and financial engineering. He is co-author (with J.Q. Hu) of the book, Conditional Monte Carlo: Gradient Estimation and Optimization Applications, which received the INFORMS College on Simulation Outstanding Publication Award in 1998. Other awards include the 1999 IIE Operations Research Division Award, a 1999 Operations Research Meritorious Service Award, a 1998 IIE Transactions Best Paper Award, and the 1995 Maryland Business School's Allen J. Krowe Award for Teaching Excellence. He is currently the Simulation Area Editor of Operations Research, and serves (or has recently served) on the editorial boards of Management Science, INFORMS Journal on Computing, IIE Transactions, and Production and Operations Management. He was also Guest co-Editor of a special issue on simulation optimization for the ACM Transactions on Modeling and Computer Simulation.

Steven I. Marcus received the B.A. degree in Electrical Engineering and Mathematics from Rice University in 1971 and the S.M. and Ph.D. degrees in Electrical Engineering from the Massachusetts Institute of Technology in 1972 and 1975, respectively. From 1975 to 1991, he was with the Department of Electrical and Computer Engineering at the University of Texas at Austin, where he was the L.B. (Preach) Meaders Professor in Engineering. He was Associate Chairman of the Department during the period 1984–89. In 1991, he joined the University of Maryland, College Park, where he was Director of the Institute for Systems Research until 1996. He is currently a Professor in the Electrical and Computer Engineering Department and the Institute for Systems Research at the University of Maryland, and Chair of the Electrical and Computer Engineering Department. Currently, his research is focused on stochastic control and estimation, with applications in semiconductor manufacturing, telecommunication networks, and preventive maintenance. He is Editor-in-Chief of the SIAM Journal on Control and Optimization.

Shalabh Bhatnagar received his Bachelors in Physics (Honors) from the University of Delhi, India in 1988, Masters and Ph.D in Electrical Engineering from the Indian Institute of Science, Bangalore, in 1992 and 1998, respectively. From Oct. 1997 to Jul. 2000, he worked as a Research Associate at the Institute for Systems Research, University of Maryland, College Park, USA. He worked as a Divisional Postdoctoral Fellow in the Division of Mathematics and Computer Science, Free University, Amsterdam, Netherlands, from Sept. 2000 to July 2001. He has been a Visiting Faculty at the Department of Computer Science and Engineering, Indian Institute of Technology, Delhi, India, since July, 2001. Starting Dec. 2001, he would be joining the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India, as an Assistant Professor.

His areas of research interest include stochastic approximation algorithms, Markov decision processes and their applications to optimization and control of discrete event systems. He is on the review committee of many journals and conferences, and in 1999 was recognized by the IEEE Transactions on Automatic Control for an outstanding job in reviewing papers.