Annals of Operations Research 28 (1991) 47–66

A SURVEY OF ALGORITHMIC METHODS FOR PARTIALLY OBSERVED MARKOV DECISION PROCESSES

William S. LOVEJOY

Graduate School of Business, Stanford University, Stanford, CA 94305-5015, USA

A partially observed Markov decision process (POMDP) is a generalization of a Markov decision process that allows for incomplete information regarding the state of the system. The significant applied potential for such processes remains largely unrealized, due to an historical lack of tractable solution methodologies. This paper reviews some of the current algorithmic alternatives for solving discrete-time, finite POMDPs over both finite and infinite horizons. The major impediment to exact solution is that, even with a finite set of internal system states, the set of possible information states is uncountably infinite. Finite algorithms are theoretically available for exact solution of the finite horizon problem, but these are computationally intractable for even modest-sized problems. Several approximation methodologies are reviewed that have the potential to generate computationally feasible, high precision solutions.

1. Introduction

A partially observed Markov decision process (POMDP) is a generalization of a Markov decision process (MDP) that allows for incomplete information regarding the state of the system. At each decision epoch, the decision maker must choose an action based only on the incomplete information at hand. Typically, one of the possible actions is to expend resources to gather additional information about the system. Thus, POMDPs are appropriate models in those contexts where actions can perform the dual role of biasing the system trajectory in some desired fashion and obtaining information that can aid in future decisions. This scenario obtains in a range of practical applications, including quality control, maintenance scheduling, military encounters, searches, data communications, statistics, health care, education, artificial intelligence and expert systems, natural resource economics, and finance. See Monahan [18] for an overview of POMDPs and some of these potential applications. There is at present a significant discrepancy between this potential and actual applied results. This is at least partially explained by the fact that, while the theoretical foundation is largely in place, analytical or algorithmic solutions for realistically sized problems are rare.

A significant body of research has been dedicated to improving our ability to solve POMDPs. One research direction is to seek problems that will have optimal value functions or policies that inherit some identifiable structure, and then exploiting that structure to accelerate the search for these optima. Some monotonicity results are available (cf. White [34,35] and Lovejoy [15]) and policy structure has been proved in some specific sequential hypothesis testing (cf. Wald [33] and DeGroot [7]) and machine maintenance (cf. Ross [26] and Rosenfield [25]) contexts, but such results are not common for POMDPs. One contributing reason is the relatively complex manner in which the state and information processes interact. Another contributing factor is a paucity of problems with known solutions, even algorithmically derived, so that an experience base that might spawn theoretical insights is largely absent.

The evolution of tractable algorithms for POMDPs is inhibited by the same dimensionality problems one faces in solving completely observed Markov decision processes. Taking cues from the relatively more advanced work for MDPs, one can expect that general algorithms will not be the means by which large POMDPs are solved. Rather, algorithmic research will make (at least) two important contributions to the general field: providing a means to numerically check hypotheses and heuristics, and suggesting what types of structural results one should look for. To emphasize this latter point, algorithmic research can suggest what specific types of structures would aid computation, and these might be different from the kinds of results one might seek in the absence of algorithmic needs. Further, the presence of tractable algorithms would give practitioners the confidence to embark upon POMDP modelling exercises with the knowledge that some solution will be forthcoming, and this in turn would encourage data collection exercises appropriate for more applications.

This paper reviews some of the currently available algorithms for the general, discrete-time, finite POMDP with the optimality criterion of maximizing the expected net present value of the induced time stream of rewards. The solution techniques are largely based on dynamic programming algorithms, and inherit their computational difficulties. Exact solution techniques are only applicable to the smallest models, but sufficiently precise approximate methods are available for some realistically sized problems.

The next section defines the finite POMDP model and reviews some known results. Section 3 reviews the methods available for solving the POMDP exactly, and section 4 reviews approximate solution methods. Concluding remarks appear in section 5.

2. Model definition

The model considered here is a discrete-time, finite partially observed Markov decision process (POMDP) with stationary cost data as in Smallwood and Sondik

[29]. In these, a decision maker chooses actions at discrete time intervals in an attempt to control the trajectory of an imperfectly observed Markovian state process. The decision maker is unable to observe the state of the system directly, but must choose his/her actions based upon information that accrues as the system evolves through time.

Let $X = \{1, 2, \ldots, n\}$ and $\Theta = \{1, 2, \ldots, m\}$ denote finite state and message sets, respectively, and let A denote a finite action set. Let $\Pi(X) = \{\pi \in \mathbb{R}^n\colon \pi \geq 0, \sum_{i=1}^n \pi_i = 1\}$, i.e., the set of probability distributions on X. Probabilistically, $\pi \in \Pi(X)$ represents a probability distribution on X. Geometrically, π is a point in the $(n-1)$-dimensional unit simplex $\Pi(X) \subset \mathbb{R}^n$. These interpretations will be used interchangeably below. The process is initiated with a known probability distribution over the state space X, $\pi_1 \in \Pi(X)$. Let $H_t = \{\pi_1, a_1, \theta_1, a_2, \theta_2, \ldots, a_{t-1}, \theta_{t-1}\}$ denote this initial distribution appended with the history of actions and messages received up to time t. At the beginning of time period t, H_t contains all the information that the decision maker can use to assess the state of the system. If, based on this information, the decision maker chooses action a_t, the following sequence of events is initiated:

(1) A real-valued reward $g(x_t, a_t)$ is received if the state of the system is x_t.
(2) The system transits to another state, x_{t+1}, in accordance with the known transition probabilities $p_{ij}^a = \Pr\{x_{t+1} = j\colon x_t = i, a_t = a\}$.
(3) A message $\theta_t \in \Theta$ is received in accordance with the known probabilities $r_{jk}^a = \Pr\{\theta_t = k\colon s_{t+1} = j, a_t = a\}$.
(4) Time increments by one, $H_{t+1} = H_t \cup \{a_t, \theta_t\}$, the decision maker must choose action a_{t+1}, and the process repeats.

Note that the reward received is not included in H_t. This does not mean that the reward is necessarily unobserved by the decision maker, since the amount received can be included implicitly in the message θ_t. The system evolves in the manner outlined above through $T \leq \infty$ time periods. If $T < \infty$ an additional salvage value $\alpha(i)$ is received at the beginning of time $T+1$ if $x_{T+1} = i$. A policy is a sequence of measurable functions $\delta_t\colon H_t \to A$ for $t = 1, 2, \ldots, T$, and the decision maker seeks a policy that maximizes the expected net present value of the time stream of rewards accrued during the process

$$E\left\{\sum_{t=1}^{T} \beta^{t-1} g(x_t, \delta_t(H_t)) + \beta^T \alpha(x_{T+1})\right\}. \tag{1}$$

$\beta \geq 0$ is an economic discount factor; if $T = \infty$ we require $\beta < 1$ and interpret $\beta^T = 0$. For a given π_1, the maximum in (1) over all policies is called the optimal value for π_1, and a policy that attains the optimal value for all $\pi_1 \in \Pi(X)$ is called an optimal policy.

It is well-known (cf. Aoki [1], Astrom [2], Bertsekas [4]) that the useful information in H_t can be encapsulated in a vector $\pi_t \in \Pi(X)$, and that the

partially observed process can be recast as an equivalent fully observed process with state space $\Pi(X)$. Specifically, let P^a denote the $n \times n$ transition matrix with components p_{ij}^a, let $R^a(\theta)$ denote the $n \times n$ diagonal matrix with element (j, j) equal to $r_{j\theta}^a$. Let e_i denote the ith unit vector in $\mathbb{R}^n$, and e an n-vector with a 1 in each component. Considering all elements of $\mathbb{R}^n$ as column vectors, define for all $\theta \in \Theta$, $a \in A$, and $\pi \in \Pi(X)$

$$\sigma(\theta;\ \pi,\ a) = \pi' P^a R^a(\theta) e, \qquad T(\pi,\ a,\ \theta) = \frac{\pi' P^a R^a(\theta)}{\sigma(\theta;\ \pi,\ a)},$$

where π' denotes π transposed. Then $\sigma(\theta;\ \pi,\ a)$ is the probability of observing θ given distribution π and action a, and by Bayes' rule $T(\pi,\ a,\ \theta) \in \Pi(X)$ is the posterior probability vector given prior probability vector π, action a, and message θ. The POMDP is equivalent, in the sense that it gives the same costs, to a sequential decision problem with "state space" $\Pi(X)$ and dynamics $\pi_{t+1} = T(\pi_t,\ a_t,\ \theta_t)$. This problem can be solved as a dynamic program. Let g^a denote the n-vector with components $g^a(i,\ a)$ and α the n-vector with components $\alpha(i)$. Define $B(\Pi(X))$ to be the set of all bounded, real-valued functions on $\Pi(X)$, and define the mapping h: $\Pi(X) \times A \times B(\Pi(X)) \to \mathbb{R}$ by $h(\pi,\ a,\ V) = \pi' g^a + \beta \Sigma_{\theta \in \Theta} \sigma(\theta;\ \pi,\ a) V(T(\pi,\ a,\ \theta))$. For $T < \infty$ (cf. Astrom [2], Bertsekas [4]) the search for an optimal policy can be restricted to Markov policies δ_t: $\Pi(X) \to A$ without loss of optimality, and the problem can theoretically be solved via the dynamic programming recursion

$$V_{T+1}^*(\pi) = \pi'\alpha \qquad \pi \in \Pi(X),$$

$$V_t^*(\pi) = \max\{h(\pi,\ a,\ V_{t+1}^*):\ a \in A\} \qquad \pi \in \Pi(X),\ t = 1, 2, \ldots, T. \quad (2)$$

V_1^* maps $\pi \in \Pi(X)$ into the optimal value for the problem initiated with distribution $\pi_1 = \pi$. Any sequence δ_t^*: $\Pi(X) \to A$ that for each t maps $\pi \in \Pi(X)$ into a maximizing argument in (2) will be an optimal policy. For $t = 2, 3, \ldots, T$, V_t^* has the interpretation of the optimal value function for the analogous problem with horizon $T - t + 1$.

For $T = \infty$ (cf. Blackwell [5]) policies can be restricted to stationary Markov policies δ: $\Pi(X) \to A$. For any V and W in $B(\Pi(X))$, define $\rho(V, W) = \sup\{\ |V(\pi) - W(\pi)| : \pi \in \Pi(X)\}$. $B(\Pi(X))$ is a complete metric space with the metric ρ, and starting with any $V_0^* \in B(\Pi(X))$ the recursion (2) will generate a sequence of value functions $\{V_t^*\} \subset B(\Pi(X))$ that will converge (in ρ) as $t \to -\infty$ to $V^* \in B(\Pi(X))$, the optimal value function for the infinite-horizon problem. Any δ^*: $\Pi(X) \to A$ that maps $\pi \in \Pi(X)$ into a maximizing action in $h(\pi,\ a,\ V^*)$ will be an optimal stationary policy for the infinite-horizon problem.

These theoretical results do not directly translate into practical solution methodologies. The fundamental problem is that at each iteration of the dynamic programming recursion, V_t^* must be evaluated at each $\pi \in \Pi(X)$, an uncountably infinite set. All feasible numerical methods implicitly or explicitly involve reducing the number of distributions π considered in each iteration to a finite

number. It is shown by Sondik [30] and Smallwood and Sondik [29] that for any finite horizon $T < \infty$ one can define the optimal value function (and hence policy) exactly on all of $\Pi(X)$ by considering only a finite number of points in $\Pi(X)$. Unfortunately, the number of points one must entertain can grow explosively in T, and the computational overhead necessary to find the "right" points can be prohibitive. As a consequence, several approximate algorithms are available, each of which explicitly bounds the calculational burden at each iteration.

3. Exact methods for finite-horizon problems

3.1. ALGORITHMS

The first exact algorithm for POMDPs was derived by E.J. Sondik in his 1971 Ph.D. dissertation [30] under the guidance of Prof. R.D. Smallwood at Stanford University. Sondik proves that for any finite t, $V_t^*(\pi)$ is piecewise linear and convex on $\Pi(X)$ (See Sawaki [28] for a generalization to piecewise linear Markov decision processes). Hence, V_t^* has a representation as the maximum of a finite number of linear functions, that is, $V_t^*(\pi) = \max\{\gamma'\pi\colon \gamma \in \Gamma_t\}$ for some finite set Γ_t of n-vectors. Elements of Γ_t will be called "gradient vectors," because any $\gamma \in \Gamma_t$ such that $V_t^*(\pi) = \pi'\gamma$ is a subgradient (cf. Rockafellar [24]) of the convex function V_t^* at the point π. Let the vectors $\gamma \in \Gamma_t$ be indexed by integers, that is $\Gamma_t = \{\gamma^1, \gamma^2, \ldots, \gamma^{\nu_t}\}$ if $\#\Gamma_t = \nu_t$, where $\#\Gamma$ denotes the cardinality of the set Γ. It is apparent that $\Gamma_{T+1} = \{\alpha\}$, and Sondik shows that given Γ_{t+1}, the expression (2) for V_t^* reduces to

$$V_t^*(\pi) = \max\left\{\pi'\left[g^a + \sum_{\theta=1}^{m} P^a R^a(\theta)\gamma^{l(\pi,a,\theta)}\right]\colon a \in A\right\}, \tag{3}$$

where $l(\pi, a, \theta)$ is the index of the maximizing γ vector in $\max\{\pi'P^aR^a(\theta)\gamma\colon \gamma \in \Gamma_{t+1}\}$, and the maximizing $a \in A$ will be an optimal action from π in time t. From this it is apparent that a candidate for inclusion in Γ_t is the vector $[g^a + \sum_{\theta=1}^{m} P^aR^a(\theta)\gamma^{l(\pi,a,\theta)}]$. Sondik presents an algorithm for finding the minimal set of candidate γ vectors necessary to represent Γ_t, and hence V_t^*. Simultaneously, Sondik's algorithm partitions $\Pi(X)$ into regions wherein a single identified action is optimal, and regions over which each γ vector is operative, that is, $V_t^*(\pi) = \pi'\gamma$ for a specified γ. Because these tasks are accomplished with a single pass over the simplex $\Pi(X)$, the algorithm is commonly referred to as Sondik's "one-pass" algorithm.

The one-pass algorithm is initiated with Γ_{t+1} and any specific $\pi_0 \in \Pi(X)$. Let $\gamma_0^a = [g^a + \sum_{\theta=1}^{m} P^aR^a(\theta)\gamma^{l(\pi_0,a,\theta)}]$, and let $\gamma_0 = \gamma_0^{a^*}$ satisfy $\pi_0'\gamma_0 = \max\{\pi_0'\gamma_0^a\colon a \in A\}$. Then γ_0 is an initial entry into Γ_t, since $V_t^*(\pi_0) = \pi_0'\gamma_0$. Sondik generates a

series of linear inequalities that define the region in $\Pi(X)$ over which γ_0 is the operative gradient vector for V_t^*, i.e., $V_t^*(\pi_0) = \pi_0'\gamma_0$. First, he notes that for any fixed a, as π moves away from π_0 the gradient vector γ_0^a will change if for any θ, $T(\pi, a, \theta)$ crosses a linear segment boundary in the representation of V_{t+1}^*, manifested by $l(\pi, a, \theta)$ changing from $l(\pi_0, a, \theta)$. If this is prevented for any a, it must be prevented for the optimal a^*. Hence, a conservative set of constraints that ensure $l(\pi, a, \theta) = l(\pi_0, a, \theta)$ for all a on the region is

$$\pi' P^{a^*} R^{a^*}(\theta)\left[\gamma^i - \gamma^{l(\pi_0, a^*, \theta)}\right] \le 0 \qquad \text{all } \theta, \text{ all } i = 1 \text{ to } \nu_{t-1},\ i \ne l(\pi_0, a^*, \theta); \tag{4}$$

$$\pi' P^{a} R^{a}(\theta)\left[\gamma^i - \gamma^{l(\pi_0, a, \theta)}\right] \le 0 \qquad \text{all } a \ne a^*, \text{ all } \theta, \text{ all } i = 1 \text{ to } \nu_{t-1},\ i \ne l(\pi_0, a, \theta). \tag{5}$$

Another way that the operative gradient can change as π moves away from π_0 is for the optimal action to change. Note that this can occur even if (4) and (5) are satisfied. Additional constraints are added to the system to account for this possibility, as well as to constrain π to be in $\Pi(X)$:

$$\pi'(\gamma_0^a - \gamma_0) \le 0 \qquad \text{all } a \ne a^*, \tag{6}$$

$$\pi \ge 0,\ \pi' e = 1. \tag{7}$$

The system of inequalities (4)–(7) defines a region contained (perhaps properly) in the region for which $V_t^*(\pi) = \pi'\gamma_0$. Several of these constraints might be superfluous. To reduce the set to the minimal number of constraints necessary to define the region Sondik suggests that each constraint in (4) through (6) be used in turn as the objective function in a maximization linear program subject to (4)–(7). If the constraint forming the objective function is binding at optimality, then that constraint forms a boundary of the region. In this case, the optimal π vector will lie on the boundary between two regions, so that the gradient vector appropriate for the adjacent region can be constructed from this optimal π. If the optimal value in the linear program is less than zero, the constraint in the objective function is never binding on the region and can be discarded.

Sondik's algorithm starts with any π_0 in $\Pi(X)$ and via the above procedure calculates γ_0 and the boundaries of the region defined by constraints (4)–(7). Each binding constraint produces another π vector to be investigated. These are added to a search table, and when the process is complete for π_0 another π vector is withdrawn from the search table and investigated in a like manner. If a region duplicates one already investigated, it is ignored. When the search table is empty, $\Pi(X)$ has been covered and the iteration is complete. Γ_t, hence V_t^*, is then completely specified, and the next iteration can begin to construct Γ_{t-1} and V_{t-1}^*.

Each point in $\Pi(X)$ has associated with it a $\gamma \in \Gamma_t$, sufficient to define the optimal value at that point, and an optimal action a^*. Sondik proves that $\#\Gamma_t$ is finite when t is finite, so that evaluation of the operative gradient at only a finite number of points in $\Pi(X)$ is sufficient to discover all of the gradients necessary

to define V_t^*. Sondik's algorithm is designed to find a sufficient finite set of π's to investigate.

It is common in the literature to reference Sondik's dissertation in parallel with Smallwood and Sondik [29]. This latter paper presents an algorithm similar to that above, except constraint set (5) is omitted. The logic is that as long as the optimal action does not change, enforced by (6), one need only check for index $l(\pi, a, \theta)$ changes at $a = a^*$. This logic is in error, as pointed out by Mukherjee et al. [20]. Constraint (6) only ensures that the optimal action does not change as long as the indices $l(\pi, a, \theta) = l(\pi_0, a, \theta)$ for all actions a, conditions that are ensured with (4) and (5) present, but not with (4) alone. Consequently, the algorithm in Smallwood and Sondik will not work as intended. For example, consider the 3-state, 4-action, 2-message machine maintenance problem in Smallwood and Sondik, with $\Gamma_{t+1} = \{(3.9269, 2.0887, 0.75), (1.3415, 1.3415, 1.3415)\}$ and $\pi_0 = (1, 0, 0)$. Here $a^* = 1$, $\gamma_0 = (4.467, 2.43, 1.0)$, and $V_t^*(\pi_0) = \pi_0'\gamma_0 = 4.467$. Constraint sets (4) and (6) will be

$$\pi'(-2.22, -0.613, 0.592) \leq 0, \tag{4a}$$

$$\pi'(0, 0, 0) \leq 0,$$

$$\pi'(-0.25, -0.25, -0.25) \leq 0, \tag{6a}$$

$$\pi'(-1.04, -0.003, 0.427) \leq 0,$$

$$\pi'(-2.54, -0.503, 0.927) \leq 0.$$

These inequalities, plus the constraints (7) that π is in the unit simplex, are not sufficient to define the region over which $V_t^*(\pi) = \pi'\gamma_0$. The vector $\pi = (0.3, 0, 0.7)$ is interior to the region defined by (4a) and (6a), yet for this π we have $a^* = 2$, $l(\pi, a^*, 1) = 1$, $l(\pi, a^*, 2) = 2$, and $V_t^*(\pi) = 2.08 > \pi'(4.467, 2.43, 1.0) = 2.04$. The problem is that $l(\pi, 2, 2) \neq l(\pi_0, 2, 2)$, a change that is prevented by including the constraints (5).

Note that any constraint of the form $\pi'y \leq 0$, where $y \in \mathbb{R}^n$ has nonpositive components, will never be violated for any $\pi \in \Pi(X)$ and can be eliminated without loss of precision (cf. Eagle [8]). For example, the second and third constraints in the set (4a), (6a) above can be eliminated.

The full set of constraints (4)–(7) are conservative in that they may define a region contained properly within the operative region for a given γ_0 vector. This means that Sondik's one-pass algorithm can generate more boundaries than necessary to define V_t^*. The relaxation effected by omitting (5) risks generating too large a region, in that some operative boundaries might be missed. Cheng [6] proposes some algorithms based on alternative relaxations of (4)–(7) wherein each vertex of each relaxed region is investigated in turn for new gradient vectors and new relaxed regions. The correct partition of $\Pi(X)$ is constructed by systematic refinement of these relaxed regions, resulting in fewer boundaries than the one-pass algorithm, and attendant computational savings. Cheng's alternative

algorithms differ primarily in the specific relaxation of (4)–(7) used. One of these, the "linear support algorithm," will be reviewed here because it can easily be modified to produce approximate solutions with error bounds, a desirable feature for the computationally complex POMDP.

$\Gamma_{T+1} = \{\alpha\}$ always. Cheng's linear support algorithm starts with Γ_{t+1}, calculates the operative gradients at the extreme points of $\Pi(X)$, and puts these in a set G_t, which approximates Γ_t. For each $\gamma_0 \in G_t$ the convex region $R(\gamma_0) = \{\pi \in \Pi(X)\colon \pi'\gamma_0 \geq \pi'\gamma \text{ all } \gamma \in G_t\}$ is constructed. If $v_t(\pi) = \max\{\pi'\gamma : \gamma \in G_t\}$ is used to approximate $V_t^*(\pi)$, the error incurred at any $\pi \in R(\gamma_0)$ will be $V_t^*(\pi) - \pi'\gamma_0$. This latter expression, being convex in π, will attain its maximum over $R(\gamma_0)$ at an extreme point of $R(\gamma_0)$. Hence, one need only check the error at all extreme points of each region to find the maximal error over all $\Pi(X)$ for using v_t instead of V_t^*. If this maximal error is zero, then $G_t = \Gamma_t$ and the iteration is complete. If the maximal error is positive, then the vertex that attains the maximal error will have an associated gradient vector not already included in G_t. This new gradient is added to G_t and the process of generating and checking extreme points is repeated. Since Γ_t is finite, the process must terminate in finite time with the appropriate Γ_t set.

Cheng builds up the Γ_t set by expanding an approximate set G_t until the error is zero. Monahan [18] starts with a $\mathcal{G}_t$ set that is quite large (although the error is already zero), and pares it down to the appropriate Γ_t set. Starting with Γ_{t+1}, Monahan suggests that one start by constructing all possible candidates for Γ_t, that is, all vectors of the form $\vartheta = (g^a + \sum_{\theta \in \Theta} P^a R^a(\theta)\gamma_\theta)$, where $a \in A$ and for each θ, γ_θ is an element of Γ_{t+1}. Denote this set by $\mathcal{G}_t$. $V_t^*(\pi) = \max\{\pi'\vartheta : \vartheta \in \mathcal{G}_t\}$ so that $\mathcal{G}_t$ could be used as Γ_t. However, $\mathcal{G}_t$ has $(\#A)(\nu_{t+1})^m$ elements, and many of these might be superfluous in defining V_t^*. A specific $\vartheta_0 \in \mathcal{G}_t$ is "dominated", and unnecessary in defining V_t^* if for all $\pi \in \Pi(X)$, $\pi'\vartheta_0 < \max\{\pi'\vartheta : \vartheta \in \mathcal{G}_t\}$, or equivalently if $\pi'\vartheta_0 < z$ for all $z \geq \pi'\vartheta$ for all $\pi \in \Pi(X)$ and $\vartheta \in \mathcal{G}_t$. This can be detected by the following linear program:

$$\begin{aligned} &\min\ z - \pi'\vartheta_0 \\ &\text{subject to}\quad z \geq \pi'\vartheta,\ \vartheta \in \mathcal{G}_t, \\ &\qquad\qquad \pi \geq 0,\ \pi'e = 1. \end{aligned} \tag{8}$$

If the optimal objective function value is positive, then ϑ_0 is dominated and can be discarded. Monahan suggests that each $\vartheta \in \mathcal{G}_t$ be tested in turn, discarded if dominated, and Γ_t is defined as the resulting reduced set of vectors. One can then proceed directly to the next iteration, constructing $\mathcal{G}_{t-1}$ and reducing this via a series of linear programs to Γ_{t-1}, etc. The optimal policies and value functions need not be calculated en route, but if at any point in time the optimal action and value for a specific π is sought, it can be easily calculated from (3).

Eagle [8] presents a modification of Monahan's method, wherein the number of constraints in the linear program (8) is reduced by one. Eagle suggests the test for

dominated vectors be performed by

$$\begin{aligned} &\min\ z - \pi'\vartheta_0 \\ &\text{subject to}\quad z \geq \pi'\vartheta,\ \vartheta \in \mathcal{G}_t,\ \vartheta \neq \vartheta_0, \\ &\qquad\qquad \pi \geq 0,\ \pi' e = 1, \end{aligned} \tag{9}$$

in which case ϑ_0 is dominated if the optimal objective function value is nonnegative. Eagle also makes the observation that if any $\vartheta \in \mathcal{G}_t$, $\vartheta \neq \vartheta_0$ satisfies $\vartheta \geq \vartheta_0$ (using the componentwise partial order) then ϑ_0 is obviously dominated and one need not construct the linear program to test this. Hence, each vector in turn is checked against the remaining set and discarded if it is majorized componentwise by any other vector. Eagle observes that this reduced the computation time in his problems, and further suggests that for small problems one might rely on the simple test for majorized vectors alone in defining Γ_t. This would probably result in a larger than necessary Γ_t set, but avoids linear programs altogether.

3.2. DISCUSSION

Two dimensions along which one might wish to compare competing algorithms are (a) computational efficiency, and (b) transparency. This latter attribute refers to the ease with which a researcher skilled in the basic techniques of Operations Research can understand and implement the algorithm. This naturally contains some subjective assessments. Computational efficiency measures, likewise, can depend on the particular machine, code and compiler used (this is particularly true when one considers the potential for optimized object codes and/or parallel processing). Subject to these caveats, however, one can say that Monahan's (and Eagle's extension) algorithm is the most transparent. This attribute can translate into more efficient code development and debugging, and higher confidence in the results. However, to generate Γ_t this algorithm requires that on the order of $(\#A)(\nu_{t+1})^m$ constraints be constructed, and a linear program be solved for each constraint. This can become prohibitive if ν_{t+1} is large. Typically only a small subset of the $(\#A)(\nu_{t+1})^m$ candidates for Γ_t will actually belong to the set, so that efficiency will be enhanced if one can build up to Γ_t instead of paring down to it. Sondik's and Cheng's algorithms are designed to do this. Of these, Cheng constructs fewer regions, which should result in more efficient execution.

This intuition is supported by some numerical trials conducted by Cheng. Fortran implementations of these algorithms were compared on the basis of cpu time on an Amdahl 5860 at the University of British Columbia. Cheng found that Monahan/Eagle consistently outperformed Sondik, and Cheng in general outperformed Monahan/Eagle. Exceptions to this latter result were when ν_{t+1} was small. $T = 20$ for all test problems, and the largest problem for which meaningful comparisons could be drawn had four core states ($n = 4$). Larger problems resulted in numerical errors, for reasons explained below.

It is reasonable to expect that none of these algorithms will find routine use in practice, depending as they do on a complete specification each time period of the set of gradients necessary to define the optimal value function. This set is finite for any finite time horizon, but its cardinality can grow exponentially in time. The optimal value function will converge as time goes to infinity to some convex, but not necessarily piecewise linear, function. Geometrically, one can visualize the number of linear segments necessary to represent the function going to infinity as time progresses. Unfortunately, an intractable number of gradients can be generated after only a few iterations in applied problems.

To compound these difficulties, the linear programs designed to identify the boundaries of regions or to reduce the set of candidate vectors rely essentially upon differences of the form $\pi'\vartheta_0 - \pi'\vartheta$ for candidate vectors ϑ_0 and ϑ. As these two candidates get closer to each other, this difference becomes increasingly difficult to distinguish from zero, resulting in precision problems in practical implementations of the algorithms on finite-precision computers. The operative gradients will get increasingly close to each other, without being equal, on any portion of $\Pi(X)$ on which the value functions V_t^* are converging to a smooth but nonlinear limit. Precision problems will arise long before any such limit is reached (this convergence of gradients may be avoided if the optimal policy is "finitely transient", see next section).

The dimensionality and precision problems are sufficiently severe to render most realistic POMDPs intractable by these exact methods.

4. Approximate algorithms for finite- and infinite-horizon problems

Results that occur in the limit as t goes to infinity are not generally realizable from a practical perspective. For this reason, one most often accepts as a solution to an infinite-horizon problem a value function (and associated policy) that is arbitrarily close to the optimal value. Each of the finite-horizon methods reviewed in the previous section can be included in the list of possible approximate algorithms for infinite-horizon models, where $\beta < 1$ is assumed. Specifically, it can be shown (cf. Heyman and Sobel [12]) that in the infinite-horizon case if $\rho(V_{t-1}^*, V_t^*) \leq \varepsilon$ then the policy found in the process of calculating V_{t-1}^* (that is, in the maximization of $h(\pi, a, V_t^*)$) will attain a value within $2\beta\varepsilon/(1-\beta)$ of the optimal value. Hence, one can find policies with values arbitrarily close to optimal by continuing iterations until $\rho(V_{t-1}^*, V_t^*) \leq \varepsilon$, for any chosen ε. Since $\{V_t^*\}$ is a Cauchy sequence in $B(\Pi(X))$, this approximation technique will terminate in finite time. This practical methodology is common in fully observed, infinite-horizon Markov decision processes.

However, some techniques for POMDPs have been developed that exploit the fact that an infinite-horizon problem will have contraction properties and an optimal stationary policy, characteristics not necessarily shared by their finite-

horizon counterparts. These techniques are specifically designed for infinite-horizon models.

The fundamental problem remains the uncountable nature of $\Pi(X)$, since only a finite representation will admit tractable algorithms in the general case (that is, without exploiting special structure). White and Scherer [36] and Cheng present some acceleration procedures to speed up convergence in infinite-horizon models, but these depend on periodic exact updates that will inherit the cardinality and precision problems mentioned above. An exact finite representation is theoretically available for the finite-horizon case, but as noted above, the cardinality of the representation can be sufficiently large to be considered "infinite" from the perspective of practical implementations.

The approximation techniques for designing a tractable, finite representation for $\Pi(X)$ fall into two main categories: finite-memory and finite-grid approximations. In the former, only a finite sequence of past actions and messages is kept by the decision maker, an effective truncation of the cardinality of H_t. Given any prior, a finite memory will induce at most a finite number of posterior points in $\Pi(X)$. In finite-grid approximations, $\Pi(X)$ is represented by a finite grid of points, directly restricting the number of π vectors considered to those points in the grid. This taxonomy is illustrative but not strict, as will be clear from the discussion.

4.1. FINITE-MEMORY APPROXIMATIONS

Sondik [31] describes an approximate method for infinite-horizon POMDPs based on Howard's [13] policy improvement algorithm. Let V_δ denote the value of a stationary policy δ in the infinite-horizon POMDP. Howard shows that the actions determined at each π in the maximization of $h(\pi, a, V_\delta)$, implemented in all time periods, define a stationary policy, call it δ', such that $V_{\delta'} > V_\delta$ if $h(\pi, \delta'(\pi), V_\delta) > V_\delta(\pi)$ for any π. The uncountable nature of $\Pi(X)$ is an impediment to this process, since V_δ need not have the structure (piecewise-linear and convex) necessary to use the tractable update (3). Sondik introduces a class of stationary policies, called "finitely transient", for which V_δ will be piecewise linear (also see Sawaki and Ichikawa [27]). If a finitely transient policy δ happens to be optimal, then $V^* = V_\delta$ will be both piecewise linear and convex. Hence, Sondik provides sufficient conditions for a finite-memory representation to be exact. Unfortunately, optimal policies need not be finitely transient in general, and the conditions for finite transience are difficult to check a priori. Sondik suggests an approximate algorithm that will discover finite transience en route, if it is present.

For an arbitrary stationary policy δ and integer $k < \infty$, V_δ can be approximated by a piecewise linear function that, in general terms, mimics finitely transient behavior for the first k time periods. As k gets larger, the approxima-

tion gets better, but the number of linear pieces in the approximation gets larger as well. Let ϑ_j denote the gradient vector associated with the jth linear piece in a k-step approximation to V_δ. Sondik defines $V_\delta'(\pi) := \max\{\pi'\vartheta_j : \text{all } j\}$ as a secondary approximate function that is both piecewise linear and convex, and hence allows the update (2) to reduce to expressions of the form (3). This is used to construct a new stationary policy, and the process is repeated. Bounds on the approximation error are calculated dynamically, and if they exceed a prescribed limit the process is repeated with an increased k. Sondik presents some examples in this paper and his dissertation [30], but the author is unaware of any further computational experience with this algorithm.

Platzman [21] suggests an approximate algorithm in which decisions are based upon a finite string of the most recent actions and observations, but notes that reasonable results may require intractably long memories. This is discussed further below. Platzman [23] generalizes the finite-memory idea by allowing for a finite number of "memory states", which will typically be some subset of the space of possible histories, e.g. a memory state might be a particular sequence of recent messages and actions, or might be an aggregation of such possibilities. Memory state transitions occur as a result of taking actions and getting messages, as do internal state transitions. Platzman presents techniques for evaluating a policy in this finite approximate system, bounding its performance relative to the optimal value, and for constructing new finite approximations. He allows for randomized policies, which are not theoretically necessary to attain optimality, observing that randomization can improve performance over nonrandomized policies when memory is truncated. The idea is that the random component represents that portion of history unknown to the decision maker. Some numerical examples drawn from Sondik's dissertation and Smallwood and Sondik [29] demonstrate the technique. The author is unaware of any further computational experience with this algorithm.

White and Scherer [37] further explore the approximation technique of truncating memory at the ω most recent actions and observations. Since there are $(m \times \#A)^\omega$ possible truncated memory vectors, it is possible to rewrite the finite-memory approximations as a dynamic program with $(m \times \#A)^\omega$ states. If the distribution $\pi_{t-\omega}$ is known, then that plus the ω actions and observations since would exactly determine the posterior π_t. However, $\pi_{t-\omega}$ depends upon the entire past history, $H_{t-\omega}$, which is not known to the decision maker basing decisions only on events in the ω most recent time periods. White and Scherer calculate worst case bounds over all possible $\pi_{t-\omega}$ (that is, the max and min over all unit probability vectors e_i, $i = 1$ to n) to generate upper and lower bounds on the optimal value function. Implementation of this technique is as easy as implementing the conventional recursion (2) with a finite state space. Action elimination procedures due to MacQueen [17] can unambiguously determine an optimal action, even in this approximate setting, if the value function bounds are sufficiently tight.

Preliminary numerical trials appear to corroborate Platzman's observation that intractably long memories will be required to generate reasonable solutions for some problems. However, excellent results can be obtained in those cases where past information diminishes rapidly in importance relative to recent events. As an extreme example, rank 1 transition matrices with $p_{ij}^a = p_i^a$ for all a, i, and j will generate posterior probability vectors π with components $\pi_j = p_j^a$, depending only on the most recent action taken. Hence, likely candidates for this approximation techniques are those problems for which some measure of the distance between the posterior vectors deriving from two different priors rapidly approaches zero as time periods progress.

White and Scherer identify such problems using a metric on $\Pi(X)$ introduced by Platzman [22];

$$d(\pi, \nu) = \max\{d_1(\pi, \nu), d_1(\nu, \pi)\}, \quad \pi \text{ and } \nu \in \Pi(X),$$

$$\text{where } d_1(\pi, \nu) = 1 - \min\left\{\frac{\pi_i}{\nu_i} : \nu_i > 0,\ i = 1 \text{ to } n\right\}.$$

White and Scherer define an ergodic coefficient for Bayesian transitions as the maximum possible distance d between posterior vectors π_t given the ω most recent actions and messages and arbitrary $\pi_{t-\omega}$. Bounds on the approximation error using truncated histories are proportional to this coefficient. This result provides a means for identifying those problems for which the finite history approximation will be efficacious.

4.2. FINITE-GRID APPROXIMATIONS

Discrete approximations to uncountable state spaces are a natural and often-used approximation technique for dynamic programs (cf. Bertsekas [4]). Applied to POMDPs, this class of approximation generates a sequence of approximate value functions by recursive application of (2) at a finite grid of points $\pi \in \Pi(X)$, thereby finessing the fundamental algorithmic problem of an uncountably infinite state space. Sondik showed that for finite horizons and any piecewise linear convex value function V_{t+1}, there will exist a finite set of points in $\Pi(X)$ sufficient to completely specify V_t. Unfortunately, finding these points is equivalent to performing a full exact update, with the attendant problems mentioned above. Finite-grid approximation techniques use a more easily generated grid of points to gain tractability, but generate approximate value functions as a result. This class of approximation methods divides roughly into two subclasses: fixed-grid and variable-grid methods. In the former, the grid is set up once at the beginning of the calculations and is not revised subsequently. In the latter, information garnered in previous iterations is used to revise the grid used in current calculations. This increases the computational overhead as the grid is revised each iteration, but promises reduced errors if the revisions are done appropriately. These methods will be reviewed in turn.

4.2.1. Fixed-grid methods

Setting up a fixed grid of points to represent an uncountably infinite state space is a common approximation technique among dynamic programming practitioners, and is one of the earliest approximation methods considered for POMDPs. Kakalik [14] and Eckles [10] use an approximate value function based upon a linear interpolation between fixed, discrete points in $\Pi(X)$ in their numerical work, but do not explore the quality of this approximation. Sondik [30] develops a bound for the approximation error using Eckles' discrete approximation.

Lovejoy [16] uses a discrete grid to generate both upper and lower bounds for the optimal value function. Lower value function bounds can be generated from any finite set of points $P \subset \Pi(X)$. The algorithm uses sets of gradient vectors ΓL_t to specify value functions $VL_t(\pi) = \max\{\pi'\gamma\colon \gamma \in \Gamma L_t\}$ as in the exact methods. However, ΓL_t is generated from ΓL_{t+1} using (3) at points in P only. The algorithm is initiated with $\Gamma L_{T+1} = \Gamma_{T+1} = \{\alpha\}$. By generating gradients at points $P \subset \Pi(X)$ only, one generates a subset of all possible gradients, and the maximum number of gradients generated is $\#P$. The maximum over a subset is guaranteed to be less than or equal to the maximum over a larger set, i.e., $VL_t \leq V_t^*$ for all t. Calculating VL_t is no more difficult than iterating on a conventional dynamic program with state space P. Approximate policies can be generated using VL_t instead of V_t^* in (3).

To generate upper value function bounds, Lovejoy uses a specific grid P designed to partition $\Pi(X)$ into sub-simplices as in Eaves [9]. Again, the dynamic programming update is applied at points in P only, and a unique, continuous piecewise-linear function, call it VU_t in time period t, can be defined by interpolating between points in P. Jensen's inequality and the known convexity of V_t^* guarantee that $VU_t \geq V_t^*$ for all t. Lovejoy uses VL_t and VU_t to generate bounds on the value loss relative to the optimal value for using the policies generated by VL_t in (3) for both finite- and infinite-horizon models. Again, action eliminations can unambiguously identify optimal actions if VL_t and VU_t are sufficiently close to each other.

The cardinality of Lovejoy's grid P is $[(M+n-1)!]/[M!(n-1)!]$, where M is the number of subintervals that each edge of the unit simplex is divided up into by the discretization procedure. $\#P$ explodes for high values of n and M, so the technique can handle only sparse meshes when n is high. The algorithm is too recent to judge its eventual acceptance as a solution methodology.

4.2.2. Variable-grid methods

Variable-grid methods allow the grid of points in $\Pi(X)$ used in the approximate algorithm to vary from one iteration to the next. If problem-specific information is used in revising the grid, these methods should generate more accurate approximations than fixed-grid methods.

Sondik and Mendelssohn [32] suggest a policy improvement procedure that investigates only those points in $\Pi(X)$ that can be reached by following a specified policy. This grid is meant to approximate the value of the policy under investigation, so that the grid changes as the policy changes from one iteration to the next. The author is unaware of any further work with this algorithm.

Cheng notes that his linear support algorithm can be stopped short of zero error, or can be stopped when the number of gradient vectors generated hits some upper limit. In either case, error bounds between the truncated process and an exact procedure are in hand. His linear support algorithm adds gradient vectors based on the vertices of the generated regions that introduce the most error into the calculations. Hence, he is essentially choosing a finite number of points in $\Pi(X)$ to investigate based upon dynamic error bound calculations. If the maximal error is stored each iteration, it is an easy matter to calculate a bound on the total error between the generated value function and the optimal value function when the algorithm terminates. Like Lovejoy's fixed-grid method, Cheng's variable-grid method is too recent to judge its eventual impact.

4.3. DISCUSSION

Tractable approximation techniques for general POMDPs depend on finite representations of infinite sets. In general terms, as the cardinality of the finite representation increases, the quality of the approximation and the calculational burden both increase. The burden, however, increases at most linearly, rather than exponentially in T, so that approximate methods can analyze problems with time horizons significantly greater than those that would be tractable with exact methods. The choice for the analyst is what type of finite representation to adopt.

The finite-memory technique of White and Scherer generates bounds from worst case priors a finite number of time periods in the past, and so will be ill-suited to problems where distant history still impacts the present. However, they will likely generate good results for problems in which distant history is largely irrelevant to current status. Recognizing which problems are good candidates for these methods is an obvious area for research. There is a close relationship between this finite approximation to POMDPs and research into contraction rates and forecast horizons in fully observed Markov decision processes (cf. Morton and Wecker [19], Federgruen et al. [11], and Bean and Smith [3]).

Finite-grid methods are natural approximation techniques, readily adopted by practitioners when approximating infinite spaces. Most often, the quality of the approximation is assumed rather than calculated because worst-case bounds are too loose to be informative. Folklore in the industry is that such techniques (including aggregation and discretization) perform well, much better than rigorously calculated worst-case bounds would suggest. This optimism rests upon an assumption that the value function is sufficiently well-behaved to lend itself to a

sparse representation. Fully observed problems now have sufficient history to render such an assumption plausible. For POMDPs, however, the set of problems solved to high precision is too spare to accept conventional wisdom without further study.

In numerical trials with Lovejoy's discrete grid, the simple heuristic of generating policies from (3) using VL_t to approximate V_t^* has worked well with even sparse grids in a series of test problems drawn from the literature. In his test problems, Cheng found that his approximate method worked better than the worst-case bounds would indicate. However, both authors used test problems that may not be representative of the spectrum of problems faced by practitioners. As with finite-memory approximations, an obvious area for research is to associate problem data with properties that lend themselves to the approximation technique.

Because discrete approximations are common for fully observed MDPs, the finite-grid methods are among the most transparent of the approximate methodologies. Fixed-grid methods are the easiest to implement, since the grid is constructed only once. Variable-grid methods increase the computational overhead by adjusting the grid each time step. However, increased accuracy should result from the informed choice of grid points. No numerical tests are yet available to compare the relative efficiency of these techniques.

5. Concluding remarks

Currently, exact algorithms for general POMDPs are intractable for all but the smallest problems, so that algorithmic solution will rely heavily on precise approximation. One of the basic trade-offs made in each of the approximate algorithms reviewed is what percentage of computational effort will be expended in designing the appropriate approximation, and what percentage to actual calculation of values and policies. The more effort one allocates to design, the more accurate the approximation should be, but less residual machine time will be available to compute the solution. A competing philosophy is to simplify the design and allocate more of the scarce machine time to actual calculation. In general terms, the Sondik and Platzman approximate algorithms are design intensive and involve a significant investment of time to understand and implement correctly, whereas the White and Scherer, and Lovejoy algorithms simplify the design and are more straightforward to implement (Cheng's methods lie somewhere in the middle). While this does not speak to the issue of relative accuracy, it might help explain why the earlier methods have not been embraced and why the more recent methods may escape the same fate. This is not at all certain, however, as these later algorithms are still too new to judge their eventual impact. One further advantage accruing to simplified algorithms is that they may be better able to exploit parallel processing capabilities.

Although some of the algorithms reviewed here expand the class of problems amenable to algorithmic solution, none are capable of solving truly large problems (an attribute inherited from, and shared with, algorithms for fully observed MDPs). Let us define a "problem of size n" as a POMDP with n partially observed states, n actions, n messages, and a finite horizon of $T = 100$ time steps. Typically a discounted infinite-horizon problem will generate high-accuracy approximate value functions within 100 time steps, so these are not excluded. A "high-accuracy" solution is a value function and policy with tight, rigorous bounds on the error relative to the optimal value. Calculations are presumed to proceed on a mainframe computer, and no special structure is assumed or exploited. Cpu times measured in minutes are assumed perfectly acceptable, and cpu times beyond several days unacceptable. Precise statements are difficult (depending as they do on specific problem, code, and machine characteristics) but one can expect exact methods to fail for numerical reasons on small problems ($n = 3$, for example) well before the 100 time step horizon is reached. Approximate methods will not fail numerically, so that feasibility is limited by the cpu time one is willing to invest. Competitive approximate methods should be readily capable of generating high-accuracy solutions for problems of size 10. Beyond this the difficulty increases exponentially. The margin of feasibility will be found at problem sizes in the 20 to 30 range, and problems of size 50 are currently infeasible. These assessments may be challenged by future research, as numerical experience with these techniques is largely absent. It is apparent, however, that heuristics and/or exploiting special structure will be the key to solving large problems. The generic algorithms surveyed here will aid in this effort by making available high-accuracy solutions to non-trivial problems, enabling numerical tests on a robust problem set.

It would be helpful to have transportable codes for some of these algorithms for use by prospective researchers. The author is currently unaware of any generally available code, but the specific authors referenced here can be contacted directly for their own versions.

This review suggests several possible directions for future algorithmic research. First, the fundamental advantage of approximate methods is their parsimonious representation of the value function. There may be a simple parametric representation for convex functions (curvature, skew, etc.) that allows the value function to be represented with just a few numbers. If an efficient update for such a representation could be derived, the class of feasible problems may be significantly expanded.

Second, specific algorithms appropriate for problems with special structure are likely to perform significantly better than generic methods. It would be a significant contribution to the current literature to survey the potential applied contexts of POMDPs and develop a taxonomy of specific problem classes characterized by some definable structure. Solution methods that exploit the special structure in a specific problem class may be quite efficient.

Third, it is important to identify specific problem classes that are amenable to heuristics. As suggested above, aggregation and discretization methods in fully observed MDPs often outperform the available theoretical bounds. Is the same true for POMDPs? Only numerical experience will develop a conventional wisdom as to what heuristics are effective.

As suggested in the introduction, computation remains an essential component of the spectrum of research in partially observed systems. This paper is intended to inform potential and practicing participants of some of the available computational options.

Acknowledgements

I would like to thank the anonymous referees for their careful reading and helpful suggestions, and Derek Ayers for his insights on relative computational burdens.

References

[1] M. Aoki, Optimal control of partially observable Markovian systems, J. Frankl. I 280 (1965) 367–386.
[2] K.J. Astrom, Optimal control of Markov processes with incomplete state information, J. Math. Anal. Appl. 10 (1965) 174–205.
[3] J. Bean and R. Smith, Conditions for the existence of planning horizons, Math. Oper. Res. 9 (1984) 391–401.
[4] D. Bertsekas, *Dynamic Programming and Stochastic Control* Academic Press, 1976).
[5] D. Blackwell, Discounted dynamic programming, Ann. Math. Stat. 36 (1965) 226–235.
[6] H. Cheng, Algorithms for partially observed Markov decision processes, Ph.D. dissertation, Faculty of Commerce and Business Administration, University of British Columbia (1988).
[7] M. DeGroot, *Optimal Statistical Decisions* (McGraw-Hill, New York, 1970).
[8] J. Eagle, The optimal search for a moving target when the search path is constrained, Oper. Res. 32 (1984) 1107–1115.
[9] B.C. Eaves, *A Course in Triangulations for Solving Equations with Deformations* (Springer, New York, 1984).
[10] J. Eckles, Optimum replacement of stochastically failing systems, Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University, Stanford CA (1966).
[11] A. Federgruen, P. Schweitzer and H. Tijms, Contraction mappings underlying undiscounted Markov decsion processes, J. Math. Anal. Appl. 65 (1978) 711–730.
[12] D. Heyman and M. Sobel, *Stochastic Models in Operations Research*, vol. 2 (McGraw-Hill, New York, 1984).
[13] R. Howard, *Dynamic Probabilistic Systems* (Wiley, New York, 1971).
[14] J. Kakalik, Optimum policies for partially observable Markov systems, Tech. report 18, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA (1965).
[15] W.S. Lovejoy, Some monotonicity results for partially observed Markov decision processes, Oper. Res. 35 (1987) 736–743.

[16] W.S. Lovejoy, Computationally feasible bounds for partially observed Markov decision processes, Research paper 1024, Stanford University Graduate School of Business, Stanford, CA (1989), to appear in Oper. Res.

[17] J. MacQueen, A test for suboptimal actions in Markov decision processes, Oper. Res. 15 (1967) 559–561.

[18] G. Monahan, A survey of partially observable Markov decision processes, Manag. Sci. 28 (1982) 1–16.

[19] T. Morton and W. Wecker, Discounting, ergodicity, and convergence for Markov decision processes, Manag. Sci. 23 (1977) 890–900.

[20] S. Mukherjee, N. Shahabuddin and K. Seth, Optimal control policies for partially observable Markov processes – A corrected and improved algorithm, unpublished manuscript, Indian Institute of Technology, Delhi, India (1985).

[21] L.K. Platzman, Finite-memory estimation and control of finite probabilistic systems, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (1977).

[22] L.K. Platzman, Optimal infinite-horizon undiscounted control of finite probabilistic systems, SIAM. J. Control Opt. 18 (1980) 362–380.

[23] L.K. Platzman, A feasible computational approach to infinite-horizon partially observed Markov decision processes, Technical note J-81-2, Georgia Institute of Technology, Atlanta, GA (1981).

[24] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, NJ, 1970).

[25] D. Rosenfield, Markovian deterioration with uncertain information, Oper. Res. 24 (1976) 141–155.

[26] S. Ross, Quality control under Markovian deterioration, Manag. Sci. 17 (1971) 587–596.

[27] K. Sawaki and A. Ichikawa, Optimal control for partially observable Markov decision processes over an infinite horizon, J. Oper. Res. Soc. Japan 21 (1978) 1–15.

[28] K. Sawaki, Piecewise linear Markov decision processes with an application to partially observable models, in: *Recent Developments in Markov Decision Processes*, R. Hartley et al. (eds.) (Academic Press, New York, 1980).

[29] R.D. Smallwood and E.J. Sondik, Optimal control of partially observable processes over the finite horizon, Oper. Res. 21 (1973) 1071–1088.

[30] E.J. Sondik, The optimal control of partially observable Markov processes, Ph.D. dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA (1971).

[31] E.J. Sondik, The optimal control of partially observable Markov processes over the infinite horizon: discounted case, Oper. Res. 26 (1978) 282–304.

[32] E.J. Sondik and R. Mendelssohn, Information seeking in Markov decision processes, Southwest Fisheries Center Administrative Resport H-79-13, Southwest Fisheries Center, National Marine Fisheries Service, NOAA, Honolulu, HI (1979).

[33] A. Wald, *Sequential Analysis*, (Wiley, London, 1947).

[34] C.C. White, Optimal control limit strategies for a partially observed replacement problem, Int. J. Sys. Sci. 10 (1979) 321–331.

[35] C.C. White, Monotone control laws for noisy, countable-state Markov chains, Eur. J. Oper. Res. 5 (1980) 124–132.

[36] C.C. White and W. Scherer, Solution procedures for partially observed Markov decision processes, Oper. Res. 37 (1985) 791–797.

[37] C.C. White and W. Scherer, Finite memory suboptimal design for partially observed Markov decision processes, Technical report, Department of Systems Engineering, University of Virginia, Charlottesville, VA (1989), submitted to Oper. Res.