

Comparison/contrast of Gomers:

more formally,
draw analogy
between Platzman
and Gomers

- ① what's relationship between M^* and set of all elements in Markov partition?
- ② what's relationship between Gomers' α and $\alpha(f)$?
- ③ when $\rho_d^n = 1 \forall n$, what does this mean?
- ④ What's difference between using $M < M^*$ and an approximate ρ ?

A Feasible Computational Approach
to Infinite-Horizon Partially-Observed
Markov Decision Problems

LOREN K. PLATZMAN
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

⑤ How to do better approximation?

Original: July, 1980
Revised: January, 1981

Submitted to Operations Research

Supported by NSF
under Grant ENG-7908952

ABSTRACT

A decision-maker, relying on incomplete and/or imperfect state observations, seeks to maximize the infinite-horizon (discounted or undiscounted) performance of an N-state Markov decision process. Since each decision depends on an infinite string of past observations and decisions (or, equivalently, on the conditional state distribution), the problem is an infinite-state Markov decision problem, i.e., an infinite-dimensional linear program. We propose to obtain suboptimal decision strategies by solving a sequence of approximations to this problem. In the M-th approximation, the decision-maker is restricted to follow a rule specified as an M-state probabilistic automaton. Each approximate problem is solved only up to a "local optimum", in simple value-iteration or policy-iteration steps. The result of each approximate problem is used to construct an initial guess for the subsequent approximate problem. Each result is also compared with an upper bound on unconstrained performance. Limited computational experience suggests that a performance near the optimum will often be obtained.

The Markov decision process (MDP) is a familiar model of operations research [7, 10, 11, 23]. The partially-observed Markov decision process (POMDP) [1, 8, 29] generalizes the MDP by requiring the decision-maker (DM) to rely on system observations, each of which is a known, stationary, possibly random function of a transition. When the states, the alternatives available to a DM before a transition, and the possible observations following a transition each form finite sets, the model is also known as a finite probabilistic system [19]. This is the case to be considered here.

As a descriptive model, the POMDP offers many advantages over the MDP, since the observations may be non-Markovian, and the decisions may affect the quality of future observations. As a prescriptive tool, however, the POMDP is awkward, for the DM may be forced to base decisions on the entire past history, an infinite string of past decisions and observations.

Early investigations of POMDP's relied on the conditional state distribution given the past history as an "external state" [1, 8]. This facilitates the analysis, but does not avoid computational difficulties, because the set of state distributions (the state space of the converted problem) is not finite.

In a finite-horizon POMDP, the summary of past decisions and observations can assume only a finite number of possible values; hence the decisions are always based on a finite-valued statistic. This fact and its implications for computing the optimal strategy were first described in a landmark paper by Smallwood and Sondik [28]. The recognition that POMDP's are not as computationally intractible as general (non-denumerable state) MDP's has led to renewed interest in the model and its applications [16].

As the horizon becomes large, the memory required to realize the optimal strategy may grow quite rapidly. In the infinite-horizon problem, symmetry guarantees that the rule for selecting decisions (based on the entire past history) will be the same at all times. Drake [8] noted that the

315

optimal decision may, in some cases, depend only on a finite string of most recent decisions and observations. The strategy is then realized by a finite-state automaton (described by a finite array of parameters) and the computational difficulties are avoided. Sondik [29, 30] has formulated a sufficient condition for the optimal strategy to take the (finite-memory) form studied by Drake, and has developed algorithms to determine the optimal strategy when it takes this form. Sondik's condition, unfortunately, is difficult to verify in practice.

Yet, in view of the fact that infinite-memory strategies are difficult to describe in a computer and difficult to implement, there is no practical alternative to finite-memory strategies, even if they fail to achieve the optimum. Thus we are led to consider computational schemes in which the strategy is restricted (in some sense) to be finite-memory. Kakalik [12] has approximated the space of internal state distributions as a finite grid. Platzman [19] allowed the DM to "forget" all but a finite string of most recent decisions and observations. In each case the DM acts on the basis of a finite-valued statistic which is a function of the past history. This statistic may also be viewed as an aggregation [25,26] of the past history into a finite number of regions. Both procedures are computationally unattractive because the memory must grow prohibitively large before reasonable decision rules are obtained. Sandell and Athans [24] considered a restricted-memory finite-horizon POMDP with time-varying decision rule.

This paper introduces a class of algorithms where the DM is restricted to base each decision on an M-valued statistic which may be a "random function" of the past history. The decision rule may also be random. Surprisingly, randomization can improve performance in this situation. Moreover, the statistic itself can be optimized (along with the decision rule)

to provide the most appropriate information. Starting with M small, a sequence of constrained problems is solved, using the solution for each problem to construct an initial guess in the following problem. The performance of each restricted design is compared with an upper bound on unrestricted performance; if they are close then we know that we have obtained a near-optimal rule. This is not guaranteed to occur, but computational results (described in this paper) are encouraging.

Why does randomization improve performance? The following example may provide some insight. Consider a totally unobserved system with states $\{1,2\}$ and decisions $\{1,2\}$, where decision i causes the system to enter state i with certainty, $i=1,2$. We receive a reward of two units whenever our decision causes the state to change, and no reward when our decision causes the state to remain the same. In the case $M=1$, the DM cannot remember any past decisions. Two unrandomized decision rules are available, each of which picks the same decision at all times. These rules yield no reward (except possibly on the first trial) and their undiscounted average return is zero. By comparison, a two-memory-state strategy, alternating between the two decisions, achieves the optimal average return of two. A randomized one-memory-state strategy that picks either decision with equal probability manages to act optimally half the time, and achieves an average return of one. The apparent effectiveness of algorithms introduced in this paper is largely attributable to the ability of small-memory randomized strategies to mimic the behavior of larger-memory deterministic strategies, as illustrated above. For further information concerning randomization in finite-memory decision schemes, see [4].

Our model, the POMDP will be precisely defined in Section 1. At first, we will confine our attention to the discounted model. In Section 2, we derive performance bounds upon which the algorithms are based. Alternative infinite-memory decision rules are discussed in Section 3. The algorithms are presented

in Section 4. The extension to undiscounted performance is given in Section 5. In Section 6, we illustrate our approach in an example drawn from [29]. Section 7 concludes the paper.

1. THE MODEL

The POMDP

The system to be controlled is characterized by the following finite sets:

$S = \{1,2,\dots,N\}$ = the set of internal states.

(1.1) $D = \{1,2,\dots,K\}$ = the set of decision alternatives.

$O = \{1,2,\dots,L\}$ = the set of observation values.

Its transition probabilities are given by an $L \times K$ array of $N \times N$ matrices having entries

(1.2) $P_{ij}(\ell|k) = \Pr$ [the system enters internal state j and emits observation ℓ | the system is presently in internal state i and has received decision k].

The initial internal state distribution is an N -vector π , having entries

(1.3) $\pi_i = \Pr$ [the initial internal state is i],

and belonging to the space of internal state distributions

(1.4) $\Pi = \{\pi \in \mathbb{R}^N \mid \pi_i \geq 0, i=1,\dots,N; \sum_{i=1}^N \pi_i = 1\}$.

This notation, also used in [18], enjoys the advantage over alternative POMDP representations that observations are permitted to depend on any or all of the following:

- The state vacated.
- The decision.
- The state entered.

Moreover, many expressions commonly associated with POMDP's can be written in matrix form. For example, if the state distribution is π , decision k is specified and ℓ is subsequently observed, then the new state distribution conditioned on information available to the DM (all past decisions and observations), readily obtained from (1.2) via Bayes' rule, may be compactly expressed as

$$(1.5) \quad T(\pi, k, \ell) = \frac{\pi P(\ell|k)}{\pi P(\ell|k)\eta} \in \Pi,$$

where η is a vertical N -vector of ones.

The Decision Strategy

The DM, at any time, is said to lie in a memory state $m \in \{1, \dots, M\}$.

The number of memory states, M , is a design parameter. At each decision epoch, the DM specifies a decision $k \in D$, receives an observation $l \in O$ from the system, and chooses the next memory state. We combine these tasks into a single operation, called an action. Each action takes the form $a = [k, m_1, m_2, \dots, m_L]$ where k is the decision specified, and m_l is the next memory state to be chosen if the observation happens to be l . Thus

$$(1.6) \quad A = D \times \{1, \dots, M\}^L$$

is the set of actions available to the DM at each decision epoch.

The finite-memory randomized strategy adopted by the DM is described by the array of probabilities

$$(1.7) \quad \phi_a^m = P[\text{the DM selects action } a \mid \text{the DM is in memory state } m].$$

Our task will be to optimize the strategy specification ϕ . In doing so, we are not only specifying the decision as a "random function" of the memory state; we are also specifying each memory state as a "random function" of the past history.

Although the dimensions of ϕ may be enormous, this will be of no practical concern to us, because the values of ϕ generated by our optimization procedures will be quite sparse. In the case of a deterministic strategy, for example, each distribution ϕ_a^m on A has exactly one positive entry. As we shall see, in Propositions 1 and 2, it is unnecessary to place more than N positive entries in any ϕ_a^m . Thus ϕ will contain at most $N \cdot M$ positive entries.

The Augmented System

Combining (1.2) and (1.6), we see that the system and DM together form a Markov chain which we call the augmented system. The state of the augmented system is a pair (i,m) , where i is an internal state and m is a memory state. Its transition matrix, when the DM chooses action a , takes the form

$$(1.8) \quad \hat{P}_{(i,m)(j,n)}^{(a)} = \sum_{\ell=1}^L P_{ij}(\ell|k) \delta_{m_\ell, n} \Big|_{[k, m_1, \dots, m_L]=a}$$

where δ is the Dirac delta function $\delta_{x,y} = 1$ if $x=y$, $\delta_{x,y} = 0$ if $x \neq y$. If actions are selected randomly according to the strategy ϕ , as defined by (1.7) then the augmented transition matrix becomes

$$(1.9) \quad \bar{P}_{(i,m)(j,n)}^{(\phi)} = \sum_{a \in A} \phi_a^m \hat{P}_{(i,m)(j,n)}^{(a)}.$$

The t -step augmented transition matrix under ϕ is represented by $\bar{P}^t(\phi)$.

The Performance Measure

Whenever the system is in internal state i , receives decision k , subsequently emits observation ℓ and enters internal state j , a finite reward $R[i,k,\ell,j]$ is obtained. The array R contains more information than we require, and our analysis will rely entirely on the array of N vectors $q(k)$, $k=1,2,\dots,K$ having entries

$$(1.10) \quad q_i(k) = \sum_{j=1}^N \sum_{\ell=1}^L R[i,k,\ell,j] P_{ij}(\ell|k)$$

= The expected reward when the system is in internal state i and receives decision k .

Also define augmented versions of (1.10)

$$(1.11) \quad \hat{q}_{(i,m)}(a) = q_i(k) \quad \left| \quad [k,m_1,\dots,m_L] = a$$

= The expected reward when the system is in augmented state (i,m) and the DM chooses action a .

$$(1.12) \quad \bar{q}_{(i,m)}(\phi) = \sum_{a \in A} \phi_a^m \hat{q}_{(i,m)}(a)$$

= The expected reward when the system is in augmented state (i,m) and the DM follows strategy ϕ .

The discounted performance measure associated with initial augmented state (i,m) and strategy ϕ is

$$\begin{aligned}
(1.13) \quad v_{(i,m)}(\phi) &= \sum_{t=0}^{\infty} \beta^t \sum_{j=1}^N \sum_{n=1}^M \bar{P}_{(i,m)(j,n)}^t(\phi) \bar{q}_{(j,n)}(\phi) \\
&= \sum_{t=0}^{\infty} \beta^t E[\text{Reward at time } t \mid \text{initial augmented state } (i,m) \text{ and strategy } \phi],
\end{aligned}$$

where the discount rate satisfies $0 \leq \beta < 1$. It is well known that $v_{(i,m)}^{\beta}$ is determined by the set of $N \cdot M$ linear equations expressed below, matrix form

$$(1.14) \quad v(\phi) = \bar{q}(\phi) + \beta \bar{P}(\phi) v(\phi).$$

We defer to Section 5 a discussion of the undiscounted case.

The Finite-Memory Optimization Problem

We may identify three phases in the determination of optimal decision strategies for POMDP's. In the design phase, we know the values of $M, P(\ell|k), q(k)$ and β , but not the initial state distribution π , and we must select ϕ and compute $v(\phi)$. In the initialization phase, we learn the value of π and select the initial memory state. In the realization phase, we obtain observations and specify decisions.

It is important to distinguish between these phases because the computational resources available during different phases may not be the same. For example, we may be willing to invest a large amount of computer time in design (which is done once for each model), but simultaneously wish to keep realization fairly simple. Alternatively, realization might be implemented by a microprocessor that can handle some computation between decisions.

Suppose that the realization will be in accordance with ϕ . Then initialization, for any π , is accomplished by selecting the initial memory state m that maximizes the expected performance $\sum_{i=1}^N \pi_i v_{(i,m)}(\phi)$. The design problem, consequently, is to select ϕ so that the performance determined by initialization will be maximal. Since we do not know the value of π , we cannot tell what that performance will be, but we can bound it from below by using the least favorable value of π . Thus, we must select ϕ to maximize the function $J(\phi)$, defined as

$$(1.15) \quad J(\phi) = \min_{\pi \in \Pi} \left\{ \max_{m=1, \dots, M} \left\{ \sum_{i=1}^N \pi_i v_{(i,m)}(\phi) \right\} \right\}.$$

In summary, we seek to solve:

DESIGN PROBLEM: Select a decision strategy ϕ so as to maximize $J(\phi)$.

Once ϕ has been selected, we await a specification of the initial internal state distribution π . Given π , we perform the

INITIALIZATION PROCEDURE: Pick the initial memory state m so that $\sum_{i=1}^N \pi_i v_{(i,m)}(\phi)$ is maximized.

Thereafter, the DM acts according to the

FINITE-MEMORY REALIZATION PROCEDURE:

- (1) Select an action $a = [k, m_1, \dots, m_L] \in A$ according to the probability distribution ϕ_a^m .
- (2) Specify decision k
- (3) Receive observation ℓ
- (4) Update the memory state $m \leftarrow m_\ell$, and return to (1).

2. PERFORMANCE BOUNDS

Bounds on Finite-Memory Performance

Let $w = [w_{(i,m)}]$ be an $N \cdot M$ -vector which we consider to be an approximation to $v(\phi)$. The MacQueen's bounds [14] on $v(\phi)$ corresponding to this approximation take the form

$$(2.1) \quad w + (1-\beta)^{-1} x_{-}(\phi, w) \leq v(\phi) \leq w + (1-\beta)^{-1} x_{+}(\phi, w)$$

where

$$(2.2) \quad x(\phi, w) = \bar{q}(\phi) + \beta \bar{P}(\phi)w - w$$

$$(2.3) \quad x_{-}(\phi, w) = \min_{(i,m)} \{x_{(i,m)}(\phi, w)\}, \quad x_{+}(\phi, w) = \max_{(i,m)} \{x_{(i,m)}(\phi, w)\}.$$

From (1.15) and (2.1) - (2.3), we obtain performance bounds

$$(2.4) \quad J_{-}(\phi, w) \leq J(\phi) \leq J_{+}(\phi, w)$$

where

$$(2.5) \quad J_{-}(\phi, w) = \tilde{J}(w) + (1-\beta)^{-1} x_{-}(\phi, w), \quad J_{+}(\phi, w) = \tilde{J}(w) + (1-\beta)^{-1} x_{+}(\phi, w)$$

$$(2.6) \quad \tilde{J}(w) = \min_{\pi \in \Pi} \left\{ \max_{m=1, \dots, M} \left\{ \sum_{i=1}^N \pi_i w_{(i,m)} \right\} \right\}.$$

Some properties of these bounds are given by the following lemmas:

Lemma 1 (Normalization). The bounds in (2.4) are unaffected by the transformation $w \leftarrow w + c\eta$ where c is a scalar constant and η is an $N \cdot M$ -vector of ones.

Proof. By (2.2), each entry of $x(\phi, w)$ decreases by $(1-\beta)c$, and by (2.6), each entry of $\tilde{J}(w)$ increases by c . Consequently $J_-(\phi, w)$ and $J_+(\phi, w)$, given by (2.5), are unaffected.

Lemma 2 (Value Iteration). The bounds in (2.4) converge monotonically to $J(\phi)$ under the iterative process $w \leftarrow u(\phi, w)$ where

$$(2.7) \quad u(\phi, w) = \bar{q}(\phi) + \beta \bar{P}(\phi)w = w + x(\phi, w)$$

Proof. By [14], the bounds in (2.1) converge monotonically to $v(\phi)$. By (1.13) and (2.5) - (2.6), the bounds in (2.4) must converge accordingly, as asserted.

Lemma 3 (Policy Iteration). Eq. (2.4) holds with strict equality when $w = v(\phi)$.

Proof. By (1.13) and (2.2), $x(\phi, w) = 0$. The desired result follows directly from (1.15) and (2.5) - (2.6).

Bounds on Infinite-Memory Performance

In a similar way, we may derive bounds on the performance of a DM who bases each decision on the internal state distribution conditioned on the entire past external history. This DM is confronted with an MDP whose (infinite) state space is Π , given by (1.4). The infinite-memory value function is the unique solution [3, 6] of the discounted Bellman's equation for discounted POMDP's [29, 30, 20]:

$$(2.8) \quad v^*(\pi) = \max_{k \in D} \{ \pi q(k) + \beta \sum_{\ell=1}^L (\pi P(\ell|k) \eta) v^*(T(\pi, k, \ell)) \}.$$

As an approximation to v^* , consider the performance of finite-memory strategy ϕ , itself approximated by w , when the initial internal state distribution is π and the initial memory state is determined by the initialization procedure of Section 1. We have

$$(2.9) \quad \tilde{v}^*(\pi, w) = \max_{m=1, \dots, M} \{ \sum_{i=1}^N \pi_i w(i, m) \}.$$

Note that $\tilde{v}^*(\pi, w)$ is a convex piecewise-linear function in π with support hyperplanes specified by w . Such functions arise in the solution of finite-horizon POMDP's [28]. *and some type of finite horizon MDP,*

To obtain MacQueen's bounds based on the approximation (2.9) to $v^*(\pi)$, we must substitute $\tilde{v}^*(\pi, w)$ for $v^*(\pi)$ in (2.8) and compare the two sides. The RHS is simplified, using (1.5), (1.8) and (1.11), to obtain

$$\begin{aligned}
 (2.10) \quad \tilde{v}^{**}(\pi, w) &= \max_{k \in D} \{ \pi q(k) + \beta \sum_{\ell=1}^L (\pi P(\ell|k) \eta) \tilde{v}^*(T(\pi, k, \ell), w) \} \\
 &= \max_{k \in D} \{ \pi q(k) + \beta \sum_{\ell=1}^L \max_{m_\ell=1, \dots, M} \{ \sum_{j=1}^N [\sum_{i=1}^N \pi_i P_{ij}(\ell)] \} \} \\
 &= \max_{a \in A} \{ \sum_{i=1}^N \pi_i y_{i,a}(w) \}
 \end{aligned}$$

A = {k, m_1, ..., m_L}

where

$$(2.11) \quad y_{i,a}(w) = \text{entry}(i, m) \text{ of } [\hat{q}(a) + \beta \hat{P}(a)w], \quad m=1$$

The bounds on $v^*(\pi)$ now take the form [3, p.237]:

$$(2.12) \quad \tilde{v}^*(\pi, w) + (1-\beta)^{-1} x_-^*(w) \leq v^*(\pi) \leq \tilde{v}^*(\pi, w) + (1-\beta)^{-1} x_+^*(w)$$

where

$$(2.13) \quad x^*(\pi, w) = \tilde{v}^{**}(\pi, w) - \tilde{v}^*(\pi, w)$$

$$(2.14) \quad x_-^*(w) = \min_{\pi \in \Pi} \{ x^*(\pi, w) \}, \quad x_+^*(w) = \max_{\pi \in \Pi} \{ x^*(\pi, w) \}.$$

The optimal infinite-memory strategy achieves a performance which, for consistency with (1.15), we define to be

(2.15)

$$J^* = \min_{\pi \in \Pi} \{v^*(\pi)\}.$$

$\tilde{v}^*(w)$

what's this?
 I think $J^* = \min_{\pi \in \Pi} \{v^*(\pi)\}$

From (2.12) - (2.14) with (2.6) and (2.9) we obtain performance bounds:

(2.16)

$$J_{-}^*(w) \leq J^* \leq J_{+}^*(w)$$

where

(2.17)

$$J_{-}^*(w) = \tilde{J}(w) + (1-\beta)^{-1} x_{-}^*(w), \quad J_{+}^*(w) = \tilde{J}(w) + (1-\beta)^{-1} x_{+}^*(w).$$

Moreover, the best infinite-memory strategy does at least as well as any finite-memory strategy,

(2.18)

$$J(\phi) \leq J^*.$$

Efficient Strategies

Let us consider the problem of maximizing the lower bound $J_-(\phi, w)$ in (2.4) with respect to ϕ , holding w fixed. Since only the second term in the definition (2.5) depends on ϕ , an equivalent problem is to maximize $x_-(\cdot, w)$. By (2.2), (1.9) and (1.12), $x(\phi, w)$ is affine in ϕ :

$$(2.19) \quad x_{(i,m)}(\phi, w) = \sum_{a \in A} \phi_a^m \xi_{i,a}^m(w)$$

$$(2.20) \quad \Rightarrow \xi_{i,a}^m(w) = \text{entry } (i,m) \text{ of } [\hat{q}(a) + \beta \hat{P}(a)w - w].$$

Thus we obtain the linear program (LP):

$$(2.21) \quad \left\{ \begin{array}{ll} \text{max:} & z \\ \text{subject to:} & z \leq \sum_{a \in A} \phi_a^m \xi_{i,a}^m(w), \quad i=1, \dots, N, \quad m=1, \dots, M \\ & \sum_{a \in A} \phi_a^m = 1, \quad \cdot \quad m=1, \dots, M \\ & \phi_a^m \geq 0, \quad a \in A, \quad m=1, \dots, M \end{array} \right.$$

This may be decomposed into M smaller LP's:

$$(2.22) \quad z = \min_{m=1, \dots, M} \{z^m\}$$

$$(2.23) \left\{ \begin{array}{l} \text{max:} \quad z^m \\ \text{subject to:} \quad z^m \leq \sum_{a \in A} \phi_a^m \xi_{i,a}^m(w) \quad i=1, \dots, N \\ \quad \quad \quad \sum_{a \in A} \phi_a^m = 1 \\ \quad \quad \quad \phi_a^m > 0 \quad a \in A \end{array} \right.$$

Definition 1. A strategy ϕ is called w-efficient if, for all $m=1, \dots, M$, the distribution ϕ^m is an optimal basic feasible solution of (2.23). It is called efficient if for some sequence w^1, \dots, w^M and all $m=1, \dots, M$, the distribution ϕ^m is an optimal basic feasible solution of (2.23), evaluated at $w = w^m$.

*Since
w is
efficient
mean*

Proposition 1. The design problem has at least one efficient solution.

Proof: $J(\cdot)$ is continuous and its domain is compact, so it achieves a maximum at some point ϕ . Let $w = v(\phi)$, and let ϕ^\dagger be a w-efficient strategy. By (2.4), $J_-(\phi^\dagger, w) \leq J(\phi^\dagger)$. Since ϕ^\dagger maximizes $J_-(\cdot, w)$ and ϕ maximizes $J(\cdot)$, it follows that $J_-(\phi, w) \leq J_-(\phi^\dagger, w) \leq J(\phi^\dagger) \leq J(\phi)$. But Lemma 3 implies $J_-(\phi, w) = J(\phi)$, so $J(\phi^\dagger) = J(\phi)$, and ϕ^\dagger is an optimal design.

Proposition 2. Any efficient strategy ϕ has at most N positive entries in each of its distributions ϕ^m , $m=1, \dots, M$.

Proof: Eliminating the redundant variable z in (2.23), we obtain an LP having only N constraints.

Consolidation of Results

Combining (2.4), (2.16) and (2.18), we obtain

$$(2.24) \quad J_-(\phi, w) \leq J(\phi) \leq J^* \leq J_+^*(w).$$

The algorithms to be discussed in Section 4 are based upon this system of inequalities. The basic idea is to iteratively increase J_- by improving one of its arguments with the other held fixed. The strategies obtained in this manner are always efficient. The upper bound J_+^* is computed for various w and the smallest value is retained. If J_- and J_+^* eventually become close, then ϕ is accordingly near-optimal.

It is worth noting why the remaining bounds J_+ and J_-^* will not be of interest to us. When J_- is maximized over w with ϕ fixed, then the bounds $J_-(\phi, w)$ and $J_+(\phi, w)$ will coincide; for the particular procedures to be used this is demonstrated by Lemmas 2 and 3. Similarly, we may relate $J_-^*(w)$ to the maximization of $J_-(\phi, w)$ over ϕ with w fixed. Use (2.9), (2.10), (2.13) and (2.20) to transform (2.14) into an LP:

$$(2.25) \quad x_-^*(w) = \min_{m=1, \dots, M} \{ \min_{\pi \in \Pi} \{ \max_{a \in A} \{ \sum_{i=1}^N \pi_i \xi_{ia}^m(w) \} \} \}$$

Since (2.25) is the dual of (2.22) - (2.23), it follows that

$$(2.26) \quad J_-^*(w) = \max_{\phi} \{ J_-(\phi, w) \}.$$

3. AN INFINITE-MEMORY STRATEGY

Before describing algorithms that determine ϕ , it is important to note that realization may actually be carried out in a more sophisticated way. The central concept in this paper is that ϕ is selected under the assumption that decisions will be made according to ϕ ; this simplifies the problem and makes it possible to select ϕ . But the information obtained during design may be used to achieve a performance better than $J(\phi)$.

Suppose that initialization, for some initial internal state distribution, has determined an initial memory state m , action a has been selected, and a new memory state m' has been entered in accordance with a . The DM may compute a new internal state distribution π' from (1.5), and repeat the initialization procedure to optimize m' . This can be repeated at every decision epoch. The resulting performance may be better but cannot be worse than that of ϕ [3, Section 5.2], but it cannot be computed exactly because the augmented system is no longer a finite-state Markov chain.

Specifically the alternative scheme is to solve the design problem and to await a specification π . Thereafter, decisions are made according to

INFINITE-MEMORY REALIZATION PROCEDURE:

- (1) Pick a memory state m so as to maximize $\sum_{i=1}^N \pi_i v_{(i,m)}(\phi)$.
- (2) Select an action $a = [k, m_1, \dots, m_L]$ according to the probability distribution ϕ_a^m .
- (3) Specify decision k
- (4) Receive observation ℓ
- (5) Update the state distribution $\pi \leftarrow T(\pi, k, \ell)$, and return to (1).

Remark: The infinite-memory realization procedure is identical to th
[28], except that the decisions may be randomized.

Remark: We might further improve performance by selecting the "best"
in step (2). This decision maximizes (2.10) with $w = v(\phi)$. To do so
advisable if the DM is a microprocessor for whom time is cheap.

4. THE DESIGN ALGORITHMS

The design problem may be tackled by a class of algorithms described in this section. These algorithms iterate on the triple (M, ϕ, w) where M is the DM memory size, ϕ is an appropriately dimensioned efficient strategy, and w is an $N \cdot M$ -vector of augmented state value approximations. In each operation, $J_-(\phi, w)$ may increase, and it cannot decrease. These operations are of three basic types, which we discuss in turn:

- Improve ϕ^m with w and the remainder of ϕ held fixed.
- Improve w with ϕ held fixed.
- Reduce or increase the number of memory states.

The best sequencing rule for these operations remains a topic for further study. We have adopted the guideline that ϕ and w should be improved as much as possible before M is increased. It is clear, however, that successive improvements in ϕ or w with the other held fixed will not necessarily lead to an optimum (over all pairs (ϕ, w) of dimension corresponding to M). We call the limit points of sequences generated in this manner local optima. At a local optimum, ϕ maximizes $J_-(\cdot, w)$, so it is w -efficient; and w maximizes $J_-(\phi, \cdot)$, so that $J_-(\phi, w) = J(\phi)$.

We have also not addressed the question of termination criterion selection, with the following exception: If new memory states cannot be added, then we know that we have achieved the unrestricted optimum performance J^* .

*is the related
of Smith's
work*

The Procedure to Improve ϕ^m

Our task here is to solve the LP (2.23). Note that, if $\min_i \{x_{(i,m)}(\phi,w)\} > x_-(\phi,w)$, then no adjustment in ϕ^m can improve $x_-(\phi,w)$ over its present value and so this operation is unnecessary.

Although this LP has $|A| = K \cdot M^L$ independent variables, it can be solved in an $(N+1) \times (N+1)$ workspace. For ease of notation, we drop the superscript m and argument w in (2.23). Let ϵ be a small positive number and set

(4.1) $c + x_-(\phi,w) - N\epsilon$

so that the optimal value of this LP is bounded below by $c + N\epsilon$. Introducing a new slack variable s_{N+1} , appropriately penalized so that it cannot appear in the optimal basis, we have

(4.2)
$$\left\{ \begin{array}{l} \text{max:} \quad z \\ \text{subject to:} \quad z \leq [c + (i-1)\epsilon] s_{N+1} + \sum_{a \in A} \xi_{i,a} \phi_a \quad i=1, \dots, N \\ \quad \quad \quad 1 = s_{N+1} + \sum_{a \in A} \phi_a \\ \quad \quad \quad s_{N+1} \geq 0; \quad \phi_a \leq 0, a \in A \end{array} \right.$$

or, equivalently,

$$\begin{aligned}
 & \text{max:} && z \\
 & \text{subject to:} && z + s_1 && + \sum_{a \in A} [c - \xi_{1,a}] \phi_a && = c \\
 (4.3) & && -s_1 + s_i && + \sum_{a \in A} [\xi_{1,a} - \xi_{i,a} + (i-1)\epsilon] \phi_a && = (i-1)\epsilon \\
 & && && s_{N+1} + \sum_{a \in A} \phi_a && = 1 \\
 & && s_i \geq 0, && i=1, \dots, N+1; && \phi_a \geq 0, a \in A
 \end{aligned}$$

This we recognize as an LP in canonical simplex form corresponding to the nondegenerate basic feasible solution $\phi_a = 0, \forall a \in A; s_i = (i-1)\epsilon, i=2, \dots, N; s_{N+1} = 1$. The simplex tableau may be written:

	s_1	...	s_{N+1}	...	ϕ_a	...	RHS	Basis
$i=1$								
\vdots								
\cdot	$\Omega = [\omega_{ij}]$				$\Gamma = [\gamma_{ia}]$		$\delta = [\delta_i]$	$[\text{NAME}_i]$
$i=N+1$								

where NAME_i identifies the basis variable (" s_j " or " ϕ_a ") whose column contains a one in row i and zeroes elsewhere. The first row describes the objective.

Initially,

(4.5) $\Omega =$

	j			
i				
		1	2...N	N+1
1		1	0	0
2				
⋮				
N		-1	I	0
N+1		0	0	1

1	c
2	
⋮	
N	(i-1)ε
N+1	1

NAME_i = "s_i", i=2, ..., N+1.

It is clear that after any number of elementary row operations (i.e., multiplying a row by a constant or adding one row to another), the identity

(4.6) $\Gamma = \Omega \Gamma^0$

will be preserved, where:

(4.7) $\Gamma^0 =$

	a		
i		... a ...	
1			
⋮			
N		$-\xi_{i,a} + [c+(i-1)\epsilon]$	
N+1		1	

Thus we need not explicitly list Γ when solving the LP represented by the simplex tableau (4.4), and any desired column of Γ may be easily computed from the current value of Ω , using (4.6).

One of the steps in performing the simplex algorithm is to select a column of the tableau (4.4) so as to minimize the first row entry. The first row of Ω is easily searched, but an enumeration of γ_{1a} for all $a \in A$ is clearly infeasible. Fortunately, there is an easier way to perform this task. Let:

$$(4.8) \quad \pi_i = \omega_{1,i} \quad i=1, \dots, N$$

Now, by (4.6) - (4.8),

$$(4.9) \quad \gamma_{1a} = \omega_{1,N+1} + \left[\sum_{i=1}^N [c+(i-1)\epsilon] \pi_i \right] - \left[\sum_{i=1}^N \pi_i \xi_{i,a} \right].$$

To minimize γ_{1a} with respect to a , it suffices to maximize $\sum_{i=1}^N \pi_i \xi_{i,a}$. Expanding ξ according to (2.20), we see that the action minimizing (4.9) is the action maximizing (2.10). The second expression of (2.10) suggests an efficient way to perform this maximization; K·L·M comparisons are required. Having decided which column of Γ we require, we may recover it using (4.6)-(4.7). Once the pivot has been performed, the additional column is no longer needed.

In summary, then, the LP (2.23) may be solved by the following procedure:

IMPLICIT SIMPLEX ALGORITHM :

- (0) Initialize c according to (4.1) and $\Omega, \delta, NAME$ according to (4.5). Go to Step (1).
- (1) (Pivot in a slack variable if possible). Set $\hat{j} = \arg \min \{\omega_{1j} : j=1, \dots, N+1\}$. If $\omega_{1\hat{j}} \geq 0$, then go to (2). Otherwise, set $\hat{i} = \arg \min \{\delta_i / \omega_{i\hat{j}} : \omega_{i\hat{j}} \neq 0, \delta_i / \omega_{i\hat{j}} \geq 0\}$. Set $NAME_{\hat{i}} = "s_{\hat{j}}"$. Divide row \hat{i} of $[\Omega, \delta]$ by $\omega_{\hat{i}\hat{j}}$. For $i \in \{1, \dots, N+1\} \setminus \{\hat{i}\}$, subtract $\omega_{ij} \cdot \text{row } \hat{i}$ from row i of $[\Omega, \delta]$. Repeat Step (1).
- (2) (Pivot in an action variable). Compute π , using (4.8). Select $a \in A$ to maximize (2.10). Compute $\gamma = [\gamma_{ia}]_{i=1}^{N+1}$ according to (4.6) - (4.7) and (2.20). If $\gamma_{1a} \geq 0$, then stop. Otherwise, set $\hat{i} = \arg \min \{\delta_i / \gamma_i : \gamma_i \neq 0, \delta_i / \gamma_i \geq 0, i=1, \dots, N+1\}$. Set $NAME_{\hat{i}} = "a"$. Divide row \hat{i} of $[\Omega, \delta]$ by $\gamma_{\hat{i}}$. For $i \in \{1, \dots, N+1\} \setminus \{\hat{i}\}$, subtract $\gamma_i \cdot \text{row } \hat{i}$ from row i of $[\Omega, \delta]$. Return to Step (1).

This algorithm will terminate after a finite number of iterations, so long as no degeneracy ($\delta_i = 0$) occurs. Upon completion, z^* will be found in δ_1 , and the solution is obtained by substituting the final values of $NAME_i$ and δ_i into statement: $NAME_i = \delta_i, i=2, \dots, N+1$; all other variables are zero.

Procedures to Restructure the Memory Space

Our most general task is to replace (M, ϕ, w) by (M', ϕ', w') , where M and M' may differ. Our procedure is roughly described as follows. Given a set of actions $A^* \subseteq A$, we add to ϕ a new memory state, for each $a^* \in A^*$, prescribing action a^* with probability one, and we remove from the remaining memory states the probabilities of prescribing an action in A^* . The number of memory states remaining is M' . From this updated strategy, we obtain w' . Finally, we compute a w' -efficient strategy ϕ' .

intuitive

Because the new performance bound $J_-(\phi', w')$ does not depend on the numbering of the memory states, we find it convenient to express w' in the form:

$$(4.16) \quad W' = \{ [w'_{(i,m)}]_{i=1}^N \mid m=1, \dots, M' \}.$$

Using this notation, our procedure takes the form:

MEMORY IMPROVEMENT PROCEDURE (GIVEN A^*):

- (1) Create a set Ψ of distributions on A . Initially, Ψ is empty.
- (2) For each $a \in A^*$, add to Ψ a distribution ψ_a , $\psi_a = 1$ if $a' = a$, $\psi_a = 0$ if $a' \neq a$.
- (3) For each $m=1, \dots, M$, perform step (3'), below
 - (3') If $\gamma = \sum_{a \in A \setminus A^*} \phi_a^m > 0$, then set $\psi_a \leftarrow \gamma^{-1} \phi_a^m$, $a \in A \setminus A^*$, and $\psi_a \leftarrow 0$, $a \in A^*$, then add ψ to Ψ .
- (4) Set $M' \leftarrow |\Psi|$.
- (5) Create $W' = \{ [\sum_{a \in A} \psi_a y_{ia}(w)]_{i=1}^N \mid \psi \in \Psi \}$. Arrange the elements of W' into an $N \cdot M'$ vector w' satisfying (4.16).
- (6) Compute ϕ' by solving the LP (2.23) for each $m=1, \dots, M'$.

We first show that J_- cannot decrease under this procedure.

Lemma 4. For any $\emptyset \subseteq A^* \subseteq A$, the w' determined by the memory improvement procedure satisfies

$$(4.17) \quad \text{conv } \{U(\phi, w)\} \subseteq \text{conv } \{W'\} \subseteq \text{conv } \{Y(w)\}$$

where

$$(4.18) \quad U(\phi, w) = \{[u_{(i,m)}(\phi, w)]_{i=1}^N \mid m=1, \dots, M\}$$

$$(4.19) \quad Y(w) = \{[y_{i,a}(w)]_{i=1}^N \mid a \in A\}$$

Proof. By (2.7), (1.9), (1.12), and (2.11),

$$U(\phi, w) = \{ \sum_{a \in A} \phi_a^m [y_{i,a}(w)]_{i=1}^N \mid m=1, \dots, M\}$$

and our construction of W' is

$$W' = \{ \sum_{a \in A} \psi_a [y_{i,a}(w)]_{i=1}^N \mid \psi \in \Psi \}.$$

Each ϕ_a^m is a mixture of distributions from step (2) with its corresponding distribution in step (3'), and each ψ_a is a distribution on A . Eq. (4.17) follows directly.

Theorem 2. If W' satisfies (4.17), then $J_-(\phi, w) \leq \max_{\phi'} \{J_-(\phi', w')\}$.

The proof of Theorem 2 is given in the appendix.

Corollary 1. For any $\emptyset \subseteq A^* \subseteq A$, the triple (M', ϕ', w') determined by the memory improvement procedure satisfies $J_-(\phi', w') \geq J_-(\phi, w)$.

Procedures to Improve w

Our task is now to update w so that $J_-(\phi, w)$ may increase and cannot decrease. By Lemma 3,^{3.14,} the largest possible value of $J_-(\phi, \cdot)$ is achieved by the policy iteration

$$(4.10) \quad w \leftarrow v(\phi),$$

but the evaluation of $v(\phi)$ according to (1.14) requires solving $N \cdot M$ simultaneous linear equations and is considered to be computationally unattractive. Following Puterman and Shin [22], a wide range of w -improving procedures may be represented by

$$(4.11) \quad w \leftarrow w + \lambda x^T(\phi, w); \quad x^T(\phi, w) = [\sum_{t=0}^T (\beta \bar{P}(\phi))^t] x(\phi, w).$$

$$x(\phi, w) = \bar{q}(\phi) + \beta \bar{P}(\phi)w - w.$$

The case $\lambda = 1, T = 0$ corresponds to conventional value iteration (2.7); the case $0 < \lambda < 1, T = 0$ represents Schweitzer's damped value iteration procedure [25]; the case $\lambda \ll 1$ describes an algorithm of Varaiya [32]. As $T \rightarrow \infty$, the case $\lambda = 1$ approaches (4.10). Following Popyack, Brown and White [21], define:

$$(4.12) \quad f(\lambda) = \min_{(i,m)} \{ \alpha_{(i,m)} + \lambda \alpha'_{(i,m)} \}$$

where

$$f(\lambda) = \min_{(i,m)} \{ \alpha_{(i,m)} + \lambda \alpha'_{(i,m)} \}$$

$$(4.13) \quad \alpha_{(i,m)} = x_{(i,m)}(\phi, w) - x_{-}(\phi, w)$$

$$(4.14) \quad \alpha'_{(i,m)} = \beta x'_{(i,m)} + (1-\beta) \cdot \min_{(j,n)} \{ x_{(j,n)}^T(\phi, w) \}; \quad x' = [\beta^T (P(\phi))^{T+1} - I] x(\phi, w)$$

Theorem 1. If w is updated according to (4.11), then $J_{-}(\phi, w)$ will increase by at least $(1-\beta)^{-1} f(\lambda)$. Moreover, $f(\cdot)$ is concave with $f(0) = 0$, and $f(\lambda) \geq 0$, $0 \leq \lambda \leq 1$.

Proof. Let $x_{-}^T = \min_{(i,m)} \{ x_{(i,m)}^T(\phi, w) \}$. By (2.2), (4.11), (4.13), and (4.14), $x(\phi, w + \lambda x_{-}^T(\phi, w)) = x(\phi, w) + \lambda \beta x' = x_{-}(\phi, w) \eta + \alpha + \lambda \alpha' - \lambda(1-\beta)x_{-}^T \eta$. So $x_{-}(\phi, w)$ increases by $f(\lambda) - \lambda(1-\beta)x_{-}^T$. By (2.6) and (4.11), $J(w)$ increases by at least λx_{-}^T . By (2.15), $J_{-}(\phi, w)$ increases by at least $(1-\beta)^{-1} f(\lambda)$. Concavity of $f(\cdot)$ and $f(0) = 0$ follow trivially from the definitions (4.12) - (4.13) and (2.2). By Lemma 2, $f(1) \geq 0$, and the remaining assertion follows from the concavity of $f(\cdot)$. □

Thus, λ in (4.11) may be an arbitrary number in $[0,1]$; it may be selected to maximize $f(\cdot)$ - by solving an LP or by performing a one-dimensional search; or it may be determined by a heuristic advocated in [21],

$$(4.15) \quad \lambda + \frac{1}{2} \max \{ \lambda \mid f(\lambda) \geq 0 \} = -\frac{1}{2} \max \{ \alpha_{(i,m)} / \alpha'_{(i,m)} \mid \alpha'_{(i,m)} < 0 \}.$$

Normalization, e.g., $w \leftarrow w - w_{(1,1)} \eta$ may reduce numerical difficulties. By Lemma 1, $J_{-}(\phi, w)$ is unaffected under this substitution.

*Note discussion w.r.t. LOR
norm extrapolations on p. 352,
Porteus; Totten O.R., #2,
March-April, 1978,*

What is the L.P.?

Remark . The condition (4.17) often occurs in the study of probabilistic automata [18]. An interesting problem associated with this condition, the state reduction problem, is to construct a set W' satisfying (4.17) with $|W'| < M$, or to show that no such set exists. The construction, if it succeeds, yields a strategy ϕ'' requiring less memory than ϕ , and whose performance is at least as good. This problem is apparently unsolved.

We now address the problem of choosing A^* . There are two considerations: to increase J_- , and to maintain M' within reasonable bounds. Define:

$$(4.20) \quad A^1 = \{a \mid \phi_a^m = 1, \text{ some } m\}$$

$$(4.21) \quad A^2 = \{a \mid 0 < \phi_a^m < 1, \text{ some } m\}.$$

Clearly, any elements of $A^1 \setminus A^2$ in A^* will have no effect on w' and may be ignored. Elements of $A^1 \cap A^2$ in A^* will not contribute to M' . So

$$(4.22) \quad M' \leq M + |A^* \setminus A^1|.$$

Thus the choice $A^* = A^1 \cap A^2$ is attractive. It cannot result in increased memory or decreased performance bound J_- , but it may reduce unnecessary randomization in ϕ .

If, however, $A^1 \cap A^2$ is empty, then this choice yields $w' = u(\phi, w)$ and the memory improvement procedure is equivalent to value iteration, followed by optimization over ϕ . A more general condition for the memory improvement procedure to be ineffective (in this manner) is given below.

Lemma 5. If $A^* \cap A^2 = \emptyset$ and

$$(4.23) \quad \sum_{i=1}^N \pi_i y_{ia}(w) \leq \tilde{v}(\pi, u(\phi, w)) \quad \forall \pi \in \Pi, a \in A^*$$

then $J_-(\phi', w') = \max_{\phi''} \{J_-(\phi'', u(\phi, w))\}$.

Proof The memory improvement procedure constructs $W' = \{[y_{ia}(w)]_{i=1}^N \mid a \in A^*\} \cup U(\phi, w)$, so $\tilde{v}^*(\pi, w') = \max \left\{ \left\{ \sum_{i=1}^N \pi_i y_{ia}(w) \mid a \in A^* \right\} \cup \{ \tilde{v}^*(\pi, u(\phi, w)) \} \right\} = \tilde{v}^*(\pi, u(\phi, w))$. Since $J_-(w'')$ depends on w'' only through $\tilde{v}^*(\cdot, w'')$, it follows that $J_-(w') = J_-(u(\phi, w))$. The assertion follows from (2.26) and the fact that ϕ' is chosen to maximize $J_-(\cdot, w')$.

In view of Lemma 5, we should choose A^* to be a nonempty subset of A^2 . If A^2 is empty, this cannot be done. In this case, ϕ is a deterministic strategy.

We may still avoid the consequences of Lemma 5 by discovering an a and π that violate (4.23). Equivalently, we seek a π for which

$$(4.24) \quad \tilde{v}^{**}(\pi, w) - \tilde{v}^*(\pi, u(\phi, w)) > 0.$$

Since \tilde{v}^{**} and \tilde{v}^* are convex, we have a difficult combinatorial problem.

5. THE UNDISCOUNTED CASE

As $\beta \uparrow 1$, $v(\phi)$ will most likely diverge, but $(1-\beta)v(\phi)$ converges [9]
to a vector of time-averaged rewards, or gains:

$$(5.1) \quad (1-\beta)v(\phi) \rightarrow g(\phi), \quad \text{as } \beta \uparrow 1.$$

Accordingly, we may modify (1.15) to obtain an undiscounted performance index:

$$(5.2) \quad G(\phi) = \min_{\pi \in \Pi} \left\{ \max_{m=1, \dots, M} \left\{ \sum_{i=1}^N \pi_i g_{(i,m)}(\phi) \right\} \right\}.$$

The bounds (2.24) now take the limiting form

$$(5.3) \quad x_-(\phi, w) \leq G(\phi) \leq G^* \leq x_+(\phi, w),$$

where w is a bounded $N \cdot M$ -vector that may be viewed as an approximation to $v(\phi)$,
and G^* is the unrestricted optimal undiscounted performance. These undiscounted
bounds are known as Odoni bounds [17].

With the exception of policy iteration (4.10), all of the procedures of
Section 4 remain meaningful and valid as $\beta \uparrow 1$. As an alternative to (4.10),
we may solve the LP:

$$(5.4) \quad \left\{ \begin{array}{ll} \text{max:} & g \\ \text{subject to:} & \bar{q}(\phi) + \bar{P}(\phi)w \geq w + g\eta. \\ & w \geq 0 \end{array} \right.$$

The w solving (5.4) will maximize $x_*(\phi, \cdot)$.

In view of Lemma 1, the constraint $w \geq 0$ does not affect the optimal value of (5.4), but somewhat reduces its complexity.

Note that we have made no assumption regarding ergodicity of the augmented system. Indeed, many interesting problems (e.g., hypothesis testing [4]) have multiple recurrent classes, and part of the DM's problem is to determine what class the system has entered. Thus, convergence as in Lemma 2, is no longer assured. The conditions for existence of stationary optimal strategies undiscounted POMDP's are rather complicated [20], and of no particular importance to us here, because our procedures are not guaranteed to find the optimum, even when it exists.

I

Before attempting to solve (4.24) in more detail, we should exhaust other opportunities to increase J_- . Thus it is appropriate to examine (4.24) under the assumption that (ϕ, w) is a local optimum. Specifically, we may assume that w takes the form of Lemma 1, so that

(A1)
$$x_-(\phi, w) = x_+(\phi, w);$$

and we may assume

(A2)
$$\phi \text{ is } w\text{-efficient.}$$

Lemma 6 Assume (A1). Then π satisfies (4.24), i.e., it violates (4.23), if and only if

(4.25)
$$x^*(\pi, w) > x_-^*(w).$$

Proof When all entries of $x(\phi, w)$ are the same, $\tilde{v}^*(\pi, u(\phi, w)) = \tilde{v}^*(\pi, w) + x_-(\phi, w)$.

Theorem 3 Assume (A2). If there exists no $\pi \in \Pi$ satisfying (4.25), then

$$J_-(\phi, w) = J(\phi) = J^*.$$

Proof By (2.26), and (A2), $J_-(\phi, w) = J_-^*(\phi, w)$. By assumption, $x^*(\pi, w) = x_-^*(w)$, $\forall \pi \in \Pi$, so $x_-^*(w) = x_+^*(w)$. By (2.16) - (2.17), $J_-^*(\phi, w) = J^* = J_+^*(\phi, w)$.

If (A1) holds, we may solve (4.24) by maximizing the L.H.S. of (4.25). In this way we evaluate $x_+^*(w)$, which enables us to obtain the upper bound $J_+^*(w)$ on unrestricted performance. And if $J_+^*(w) = J_-(\phi, w)$, we know that we have obtained the global optimum performance. Otherwise, we may use π to obtain A^* and apply

the memory improvement procedure.

The optimization of $x^*(\pi, w)$ over π may be accomplished in various ways. We know that $(\pi, \tilde{v}^*(\pi, w))$ must be a vertex of the polyhedron $\{(\pi, z) \mid z \geq \tilde{v}^*(\pi, w), \pi \in \Pi\}$. So we may enumerate these vertices [15] and evaluate (4.25) for each one. Or, alternatively, we may maximize the convex function $\tilde{v}^{**}(\pi, w)$ over each polyhedron $\{\pi \in \Pi \mid \sum_{i=1}^N \pi_i w(i, m) \geq \tilde{v}^*(\pi, w)\}, m=1, \dots, M$, as in [31]. The entire computation is necessary only if we desire $J_+^*(w)$. Otherwise, it may be terminated as soon as a π satisfying (4.25) has been discovered.

In summary, we have:

PROCEDURE TO SELECT A^* :

- (1) If $A^1 \cap A^2$ is nonempty, then set $A^* \leftarrow A^1 \cap A^2$.
In this case, M cannot increase.
- (2) Otherwise, if A^2 is nonempty, then set $A^* \leftarrow \{a^*\}$, where a^* is an arbitrary element of A^2 .
- (3) Otherwise, find a π satisfying (4.25), let a^* achieve the maximum in (2.10), and set $A^* \leftarrow \{a^*\}$.
- (4) If (A2) is satisfied and no π may be found in (3), then ϕ is optimal.

6. NUMERICAL EXAMPLE

Consider the following undiscounted, 2-state POMDP drawn from the doctoral dissertation of E. Sondik [29].

A wealthy industrialist employs either of two analysts to manage his holdings during any month. The holdings may be in a loss state (1) or profit state (2). Each analyst's effect on the holdings is modeled as a Markov chain with transition matrix $P(k)$ and profits $q(k)$. At the end of the month, the analyst holding the funds reports on their state. The report is correct with a probability that depends on the true state and the analyst making the report. The problem parameters take the form

$$\begin{aligned}
 (6.1) \quad P(1|1) &= \begin{bmatrix} .48 & .04 \\ .30 & .10 \end{bmatrix}, & P(1|2) &= \begin{bmatrix} .45 & .20 \\ .36 & .24 \end{bmatrix}, \\
 P(2|1) &= \begin{bmatrix} .32 & .16 \\ .20 & .40 \end{bmatrix}, & P(2|2) &= \begin{bmatrix} .05 & .30 \\ .04 & .35 \end{bmatrix}, \\
 q(1) &= \begin{bmatrix} -4 \\ 4 \end{bmatrix}, & q(2) &= \begin{bmatrix} 0 \\ 3 \end{bmatrix}.
 \end{aligned}$$

Let us start with $M=1$. For our initial guess, we select Analyst 1 at all times:

$$(6.2) \quad \phi = \begin{array}{|c|c|c|} \hline m & a & \text{Prob} \\ \hline 1 & [1,1,1] & 1 \\ \hline \end{array}$$

Solving (5.4), we see that $x_-(\phi, \cdot)$ is maximized when

$$(6.3) \quad w = \begin{array}{|c|c|} \hline i \backslash m & 1 \\ \hline 1 & 0 \\ \hline 2 & 11.429 \\ \hline \end{array}$$

Now

$$(6.4) \quad x_-(\phi, w) = -1.714.$$

We next maximize x_- with respect to ϕ . Let $\epsilon=.01$. By (6.4) and (4.1

$$(6.5) \quad c = -1.734.$$

Initially, by (4.5),

$$(6.6) \quad [\Omega, \delta, \text{NAME}] = \left[\begin{array}{ccc|c|c} 1 & 0 & 0 & -1.734 & \text{obj} \\ -1 & 1 & 0 & .01 & s_2 \\ 0 & 0 & 1 & 1 & s_3 \end{array} \right]$$

The top row of Ω is nonnegative, so we proceed to step (2). With $\pi=(1,0)$, we maximize (2.10). There are only two actions in A, and the optimum in (2.10) is achieved by $\hat{a}=[2,1,1]$. Now

$$(6.7) \quad \xi_{\cdot, \hat{a}} = \begin{pmatrix} 5.714 \\ -1.572 \end{pmatrix}, \quad \gamma_{\cdot, \hat{a}}^0 = \begin{pmatrix} -7.448 \\ -0.152 \\ 1 \end{pmatrix}, \quad \gamma_{\cdot, \hat{a}} = \begin{pmatrix} -7.448 \\ 7.296 \\ 1 \end{pmatrix}.$$

Combining (6.6) and (6.7), our workspace becomes

$$(6.8) \quad [\gamma, \Omega, \delta, \text{NAME}] = \left[\begin{array}{ccc|ccc|c} -7.448 & 1 & 0 & 0 & -1.734 & \text{obj} \\ 7.296 & -1 & 1 & 0 & .01 & s_2 \\ 1 & 0 & 0 & 1 & 1 & s_3 \end{array} \right].$$

Pivot on row 2 of γ to obtain

$$(6.9) \quad [\gamma, \Omega, \delta, \text{NAME}] = \left[\begin{array}{ccc|ccc|c} 0 & -.020 & 1.020 & 0 & -1.727 & \text{obj} \\ 1 & -.137 & .137 & 0 & .001 & \phi[2,1,1] \\ 0 & .137 & -.137 & 1 & .999 & s_3 \end{array} \right].$$

Discard γ and pivot on $\omega_{3,1}$ to obtain

$$(6.10) \quad [\Omega, \delta, \text{NAME}] = \left[\begin{array}{ccc|ccc|c} 0 & 1 & .146 & -1.581 & \text{obj} \\ 0 & 0 & 1 & 1 & \phi[2,1,1] \\ 1 & -1 & 7.299 & 7.292 & s_1 \end{array} \right].$$

This tableau is optimal, and it prescribes a new strategy:

(6.11) $\phi =$

	m	a	Prob
1		[2,1,1]	1

Solving (5.4), we see that $x_-(\phi, \cdot)$ is now maximized by

(6.12) $w =$

	m	1
1		0
2		3.333

and

(6.13) $x_-(\phi, w) = 1.667.$

This is a substantial improvement over (6.4).

As it turns out, (6.11) - (6.13) describe a local optimum. We now compare the performance (6.13) with the upper bound $x_+^*(w)$ on unconstrained performance. Since $\tilde{v}^*(\pi, w) = \pi \begin{pmatrix} 0 \\ 3.333 \end{pmatrix}$ is linear in π and $\tilde{v}^{**}(\pi, w) = \max \left\{ \pi \begin{pmatrix} -3.333 \\ 5.666 \end{pmatrix}, \pi \begin{pmatrix} 1.666 \\ 5.000 \end{pmatrix} \right\}$ is convex in π , the maximum of $x^*(\pi, w)$ occurs at (1,0) or (0,1). Thus $x_+^*(w) = 2.333$. So we now know that the infinite-memory realization of Section 3, based upon (6.12), achieves a performance of at least 1.667, and the optimal performance is at most 2.333.

I

We next seek to improve the memory space. Since $x^*(\pi, w)$ was maximized at $\pi = (0,1)$, and $\tilde{v}^{**}(\pi, w)$ is maximized in (2.10) by $a = [1,1,1]$, we take $A^* = \{[1,1,1]\}$. Now the memory improvement procedure produces:

(6.14) $w =$

	m		
i		1	2
1		-3.333	1.666
2		5.666	5.000

The w-efficient strategy is

(6.15) $\phi =$

m	a	Prob
1	[1,1,1]	1
2	[2,1,1]	1

Solving (5.3), we obtain

(6.16) $w =$

	m		
i		1	2
1		0	5
2		9	8.333

Once again, (6.15) - (6.16) is a local optimum with $x_-(\phi, w) = 1.667$, but the new upper bound is 1.697. The details of these computations are not shown.

So we now know that the strategy (6.11) achieves a performance of at least 1.667, and the optimal performance is at most 1.697. Since (6.11) achieves a performance within 2% of the optimum, we may choose to adopt strategy (6.11) - use Analyst 2 at all times - and abandon further computation.

If we choose to continue, we eventually obtain the unconstrained optimal strategy, which requires 3 memory states:

(6.17) $\phi =$

m	a	Prob
1	[2,1,2]	1
2	[2,1,3]	1
3	[1,1,1]	1

The optimal performance is 1.672.

Alternative methods for solving POMDP's might obtain the optimum, (6.17) But only our method would identify the near-optimal and less complex strategy (6.11).

1

7. CONCLUDING REMARKS

The preceding sections have introduced a novel formulation of steady-state POMDP's. Many of the central ideas, however, may be found elsewhere in the literature.

The distinction between design-time and real-time computation is a standard notion in control engineering, where the realization is performed by an electronic circuit. With the advent of microprocessors, this distinction has become somewhat blurred.

The solution of large MDP's by state aggregation methods is currently an area of intense investigation [26,27]. Our problem may be viewed as an infinite state MDP for which aggregation is a practical necessity. Approximating complex decision strategies by randomized finite-memory procedures is a familiar method in hypothesis-testing and statistical decision theory [4].

A control problem in which decisions are restricted to depend on anything less than the entire past history of decisions and observations is said to have non-classical information pattern - or simply to be a non-classical control problem. Conventional dynamic programming techniques (where decisions depend on a "state") cannot be applied in the nonclassical case. Levine, Johnson and Athans [13] consider the nonclassical problem of controlling an N-dimensional linear system where the controller is restricted to be an M-dimensional linear system. They obtain a function of two variables which is easily minimized with respect to either of its arguments with the other held fixed, but for which global minimization is a difficult combinatorial problem. As in the case of $J_-(\phi, w)$, one of these arguments represents the past (i.e., ϕ defines the memory state as a function of past history) and the other describes the future (i.e., w specifies expected future rewards as a

function of the present augmented state). Another nonclassical problem having structure similar to ours is a radar design problem involving two players: a transmitter and a receiver [5].

Our formulation has the special advantage of requiring very little computer memory, even at design time. This suggests the possibility of implementation on small machines, perhaps in an interactive mode. The value-iteration procedure was implemented on a Radio Shack TRS-80. The numerical example of Section 6 was solved in about five minutes. The machine maintenance and repair problem used as an illustrative example by Smallwood and Sondik [28] was solved in about 46 hours. Within 12 hours, it had devised a strategy whose performance was known to be within 10% of the optimum. It is not clear how much time the calculation would have required on a fast machine. But the cost of 46 hours on the TRS-80, a machine that sells for under a thousand dollars, compares favorably with the 50 seconds on an IBM 360/67 reported by Smallwood and Sondik; and they were unable to obtain the entire steady-state solution.

APPENDIX

Proof of Theorem 2

We construct a strategy ϕ' (not necessarily efficient) so that $J_-(\phi', w') \geq J(\phi, w)$.

By assumption, each element of $U(\phi, w)$ may be expressed as a mixture of elements of W' , so there is an $M \times M'$ matrix ρ such that

$$(A.1) \quad \left\{ \begin{array}{ll} u_{(i,m)}(\phi, w) = \sum_{m'=1}^{M'} \rho_{m,m'} w'(i, m'), & i=1, \dots, N, \quad m=1, \dots, M \\ \sum_{m'=1}^M \rho_{m,m'} = 1, & m=1, \dots, M \\ \rho_{m,m'} \geq 0, & m'=1, \dots, M', \quad m=1, \dots, M \end{array} \right.$$

Similarly, let ψ be an $M' \times |A|$ matrix such that

$$(A.2) \quad \left\{ \begin{array}{ll} w'(i, m') = \sum_{a \in A} \psi_{m',a} y_{i,a}(\phi, w), & i=1, \dots, N, \quad m'=1, \dots, M \\ \sum_{a \in A} \psi_{m',a} = 1, & m'=1, \dots, M \\ \psi_{m',a} \geq 0, & a \in A, \quad m'=1, \dots, M \end{array} \right.$$

Let ϕ' be the following strategy: The DM, when in "new memory state" m' chooses action $a = [k, m_1, \dots, m_L]$ with probability $\psi_{m',a}$. This action prescribes an "old memory state" $n = m_\ell$ if observation ℓ is subsequently received. The DM then selects an updated "new memory state" n' with probability $\rho_{n,n'}$. We may associate with this strategy a stochastic matrix

$$(A.3) \quad Q_{(i,m')(j,n)} = \sum_{a \in A} \psi_{m',a} \hat{P}_{(i,\cdot)}(j,n)(a)$$

whose entries represent the probabilities that this DM, starting in "new augmented state" (i, m') will undergo a transition to "old augmented state" (j, n) before selecting an updated "new memory state". Straightforward manipulations of (2.2) and (A.1) - (A.3) now yield

$$(A.4) \quad x(\phi', w') = \beta Q x(\phi, w).$$

By (2.5),

$$(A.5) \quad J_-(\phi', w') = \tilde{J}(w') + (1-\beta)^{-1} x_-(\phi', w')$$

From (2.6) and (A.1), it follows that

$$(A.6) \quad \tilde{J}(w') \geq \tilde{J}(u(\phi, w))$$

By (2.7) and (2.3)

$$(A.7) \quad \tilde{J}(u(\phi, w)) \geq \tilde{J}(w) + x_-(\phi, w)$$

Since Q is a stochastic matrix, (A.4) implies

$$(A.8) \quad x_-(\phi', w') \geq \beta x_-(\phi, w).$$

Combining (A.5) - (A.8), we obtain

$$(A.9) \quad J_-(\phi', w') \geq \tilde{J}(w) + (1-\beta)^{-1} x_-(\phi, w) = J_-(\phi, w).$$

REFERENCES

- [1] Astrom, K.J., 1965. "Optimal Control of Markov Processes with Incomplete State Information," J. Math. Anal. Appl. 10, 174-205.
- [2] Astrom, K.J., 1969. "Optimal Control of Markov Processes with Incomplete State Information II: The Convexity of the Loss Function," J. Math. Anal. Appl. 26, 403-406.
- [3] Bertsekas, D., 1976. Dynamic Programming and Stochastic Control, Academic, New York.
- [4] Cover, T.M. and Hellman, M.E., 1970. "The Two-Armed-Bandit Problem with Time-Invariant Finite Memory," IEEE Trans. Inf. Th., IT-16, 185-195.
- [5] Delong, D.F. and Hofstetter, E.M., 1967. "On the Design of Optimum Radar Waveforms for Clutter Rejection", IEEE Trans. Inf. Th., IT-13, 454-463.
- [6] Denardo, E.V., 1967. "Contraction Mappings in the Theory Underlying Dynamic Programming", SIAM Rev. 9, 165-177.
- [7] Derman, C., 1970. Finite State Markovian Decision Processes, Academic, New York.
- [8] Drake, A.W., 1962. "Observation of a Markov Process through a Noisy Channel," Sc.D. Thesis, Department of Electrical Engineering, M.I.T., Cambridge, Massachusetts.
- [9] Flynn, J., 1976. "Conditions for the Equivalence of Optimality Criteria in Dynamic Programming," Ann. Stat. 4, 936-953.
- [10] Howard, R.A., 1960. Dynamic Programming and Markov Processes, M.I.T. Press, Cambridge, Massachusetts.
- [11] Howard, R.A., 1971. Dynamic Probabilistic Systems, Vols. I and II, Wiley, New York.
- [12] Kakalik, J.S., 1965. "Optimum Policies for Partially Observable Markov Systems," M.I.T. M.S. Thesis; also M.I.T. Operations Research Center Technical Report No. 18.
- [13] Levine, W.S., Johnson, T.L., and Athans, M., 1971. "Optimal Limited State Variable Feedback Controllers for Linear Systems", IEEE Trans. Aut. Contr., AC-16, 785-793.

- [14] MacQueen, J.B., 1966. "A Modified Dynamic Programming Method for Markovian Decision Problems," J. Math. Anal. Appl. 14, 38-43. ✓
- [15] Mattheiss, T.H. and Rubin, D.S., 1980. "A Survey and Comparison of Methods for Finding All Vertices of Convex Polyhedral Sets," Math. Oper. Res. 5, 167-185.
- [16] Monahan, G., 1978. "A Survey of Partially-Observed Markov Decision Processes: Theory Models and Algorithms," to appear.
- [17] Odoni, A.R., 1969. "On Finding the Maximal Gain for Markov Decision Processes," Opns. Res. 17, 857-860.
- [18] Paz, A., 1970. Introduction to Probabilistic Automata, Academic, New York.
- [19] Platzman, L.K., 1977. "Finite-Memory Estimation and Control of Finite Probabilistic Systems," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, M.I.T.; also M.I.T. Electronic Systems Laboratory Technical Report ESL-R-723, Cambridge, Massachusetts.
- [20] Platzman, L.K., 1980. "Optimal Infinite-Horizon Undiscounted Control of Finite Probabilistic Systems," SIAM J. Contr. Opt. 18, 362-380.
- [21] Popyack, J.L., Brown, R.L., and White, C.C., 1979. "Discrete Versions of an Algorithm Due to Varaiya," IEEE Trans. Auto. Control, AC-24, 503-504.
- [22] Puterman, M.L. and Shin, M.C., 1978. "Modified Policy Iteration Algorithms for Discounted Markov Decision Problems", Management Sci. 24, 11247-1137. ✓
- [23] Ross, S.M., 1979. Applied Probability Models with Optimization Applications, Holden-Day, San Francisco, Calif.
- [24] Sandell, N.R. and Athans, M., 1974. "A Finite-State Finite-Memory Minimum Principle," J. Opt. Th. Appl. 25.
- [25] Schweitzer, P.J., 1971. "Iterative Solution of the Functional Equations of Undiscounted Markov Renewal Programming," J. Math. Anal. Appl. 34, 495-501.
- [26] Schweitzer, P.J., Puterman, M. and Kindle, K., 1981. "Iterative Aggregation/Disaggregation Methods for Discounted Markov Decision Processes" (forthcoming).
- [27] Schweitzer, P.J. and Kindle, K., 1981. "Iterative Aggregation/Disaggregation Methods for Undiscounted Markov Decision Processes" (forthcoming).
- [28] Smallwood, R.D. and Sondik, E.J., 1973. "The Optimal Control of Partially-Observable Markov Processes over a Finite Horizon," Opns. Res. 21, 1071-1081.

- 29] Sondik, E.J., 1971. "The Optimal Control of Partially-Observable Markov Processes," Ph.D. Thesis, Department of Engineering Economic Systems, Stanford University; also Stanford Information Systems Laboratory Technical Report No. 6252-4.
- 30] Sondik, E.J., 1978. "The Optimal Control of Partially-Observable Markov Processes Over the Infinite Horizon: Discounted Costs," Opns. Res. 26, 282-304.
- 31] Thoai, N.V. and Tuy, H., 1980. "Convergent Algorithms for Minimizing a Concave Function," Math. Oper. Res. 5, 556-566.
- 32] Varaiya, P., 1978. "Optimal and Sub-Optimal Stationary Controls for Markov Chains," IEEE Trans. Auto. Control, AC-23, 388-94.