ABSTRACT DYNAMIC PROGRAMMING MODELS UNDER COMMUTATIVITY CONDITIONS

SERGIO VERDI,⁷ AND H. VINCENT POOR²

Abstract: The unifying purpose of the abstract dynamic programming models is to find sufficient conditions on the recursive definition of the objective function that guarantee the validity of the dynamic programming iteration. This paper presents backward, forward, and forward-backward models that require previous sufficient conditions and that include, but are not restricted to, optimization problems. The backward-forward model is devoted to the simultaneous solution of a collection of interrelated sequential problems based on the independent computation of a cost-to-minimize function and a cost-to-go function. Several extremization and nonextremization problems illustrate the applicability of the proposed model.

Key words: backward and forward dynamic programming operator models, finite and infinite horizon sequential optimization, discrete-time Markov processes, fixed-interval detection and smoothing

AMS(MOS) subject classifications. 90C79, 90C48, 93E20

1. Introduction. The fact that dynamic programming has found application in a wide variety of sequential optimization problems has led several researchers to investigate what class of objective functions can be optimized by dynamic programming. The unifying purpose of the abstract dynamic programming models is not to facilitate the solution of specific problems, but to extract the essential features that guarantee the solvability of a problem by dynamic programming. In the models proposed by Mitten [1], Denardo [2], Nemhauser [3], Karp and Held [4] and Bertsekas [5] it is assumed that the real-valued objective function can be defined recursively by a generating operator or local income function [6] which maps a set of functions of states into itself. If this operator is monotone, then further restrictions such as the contraction-mapping assumption of [2], the continuity conditions of [5] or the finiteness of the state-space [4] suffice to validate the dynamic programming solution (a variational finite and infinite horizon settings. Also, Brown and Strauch [7] have proposed a related abstract model) wizere the return space is not necessarily the extended real line but a multiplicative lattice.

In this paper we propose an abstract discrete-time dynamic programming model that includes, but is not restricted to, optimization problems. Any functional satisfying a certain commutativity condition with the generating operator (which, unlike previous models, is not restricted to be monotone) of the objective function results in a sequential problem solvable by a dynamic-programming iteration. Examples of sequential nonextremization problems fitting this framework are the derivation of marginal probability distributions from decomposable joint distributions, iterative computation of stage-separated functions taking values on additive commutative semigroups with distributive products, generation of symbolic transfer functions, and the computation of unconditional transition probabilities of a Markov process.

Another feature of the framework of this paper is the ability to formulate forward models completely symmetric to backward ones. This enables the analysis of open-loop problems by either approach under the same kind of restrictions (§3) and, more significantly, it allows for the formulation of the backward-forward dynamic-programming operator model.

* Received by the editors July 29, 1985; accepted for publication (in revised form) June 4, 1986. This work was supported in part by the U.S. Office of Naval Research under contract N00014-82-K-0814 and the National Science Foundation under grants ECS-82-12314 and ECS-85-04752.
1 Department of Electrical Engineering, Princeton University, Princeton, New Jersey 08544.
6 Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.
7 Stage indices are omitted here.

The backward-forward model p
ABSTRACT DYNAMIC PROGRAMMING MODELS

The backward-forward model presented in this paper is devoted to the simultaneous solution of a collection of interrelated sequential problems based on the independent computation of a cost-to-arrive function and a cost-to-go function. To fix ideas, consider the following simple example of this method. In a layered network, the shortest path containing a particular arc can obviously be obtained by deleting all other arcs in the same layer and solving for the shortest route by either forward or backward dynamic programming; however, if the problem must be solved for each arc in the network, then rather than repeating the whole process, it is more efficient to simply compute the distances of each node from the source and to the destination. Other problems such as fixed-interval minimum error probability detection in data communications and fixed-intersect smoothing are shown to fit into this framework.

In §2, we present a single operator model for discrete-time finite-horizon backward and forward problems, and a pair of sufficient conditions, viz., the decomposability of the objective function and the commutativity of operators, are shown to ensure the validity of the dynamic programming iteration. When applied to initialization problems, the commutativity condition is weaker and not more difficult to check in specific problems than the sufficient conditions imposed by previous models. The use of the proposed model is briefly illustrated in §2 for optimum stochastic control problems, and in §3 for a variety of deterministic problems with extremization and nonextremization operators. Familiarity with previous abstract dynamic programming models, in particular with the model due to Bertsekas [5] which encompasses most other previous settings, may be advantageous in reading §§2 and 3. The formulation and some applications of the backward-forward model are presented in §4. Finally, §5 briefly discusses the issues arising in the infinite-horizon problem and sufficient conditions on the commutativity of operators and interchangeability of limits are shown to ensure the validity of the dynamic programming iteration and the fixed-point property of the sought-after function of states.

2. Abstract finite-horizon dynamic programming operator model.

Glossary of notation. The following notation is used throughout the paper.

S: state space.
A: action space.
μ: function mapping S to A; μ : S → A.
M: set of admissible policies from S to A; μ ∈ M.
Q: return space.
L: Q-valued function of states; L : S → Q.
H: operator mapping Q-valued functions of states to Q-valued functions of actions; H : Q → Q^M
H(a): the function of actions H:L(A)→Q evaluated at the point a ∈ A.
V: operator (functional) mapping Q-valued functions of M to Q; V : Q → Q^M.
V(μ|μ): image in Q of a function 4 : M → Q; note that μ is a dummy argument: V(μ|μ) depends on the values of g(μ) for all μ ∈ M.
V(L): Q-valued function of states whose value at x ∈ S is V(L(x)), where L is a Q-valued function of states parameterized by μ ∈ M. Analagous notation is used for functions of actions V(HEL).

a 1Stage indices are omitted here.
\[ V_{\mu}(x) = V_{\mu}(y) \] for all \( x, y \in S \).

\[ P_{\mu} : \text{projection of } \mu \text{ on } \text{Cartesian product } \mu(x_1, \ldots, x_n) = x_i; \]
\[ P_{\mu} : \text{projection of } \mu \text{ on } x_i = x_i. \]

The foregoing result states that the action spaces and operators \( H \) and \( V \) take on various meanings depending on the specific problem to which the model is applied. In optimization problems, \( V \) is identified with the extended real line and \( V = \Delta \) is the infinitesimal functional; i.e., for any function \( q : M \rightarrow Q \)
\[ V(q(\mu)) = \inf_{\mu \in \Delta} q(\mu). \]

To fix ideas in a first reading of the sequel, it may be helpful to identify \( Q \) and \( V \) with these particular cases.

For \( i = 0, \ldots, N \), let \( M \) be the set of admissible policies mapping \( S_i \rightarrow A_i \). The state and action spaces at stage \( i \) (respectively). Suppose that for each stage \( s \in [0, \ldots, N] \), we are given a function of states \( L_{\mu}(x) \rightarrow Q \), which is parameterized by the string of policies \( \mu_{1:s} \in M_{1:s} \). \( L_{\mu}(x) \) is the cost incurred by using the policies \( \mu_{1:s} \) if \( x \) is the state at the \( n \)th stage. Then the objective of this section is to investigate under what conditions dynamic programming can be used to find the function of initial states:
\[ V_{\mu}(L_{\mu}(x)) \]

where \( V_{\mu} : Q^N \rightarrow Q, i = 0, \ldots, N. \)

The first result is to order to solve (1) by dynamic programming is the recursive formulation of the sequential dependence of the function of states \( L_{\mu}(x) \) on the polices \( \mu_{1:s} \).

**Property 1.** Backward decomposability of the objective function. There exists a collection of operators \( H^i : Q^N \rightarrow Q^N, i = 1, \ldots, N, \) such that for all \( \mu_{1:N} \in M_{1:N} \)
\[ L_{\mu}(x) = H_{\mu}(L_{\mu}(x)), \quad x \in S_{1:s}, \quad i = 1, \ldots, N. \]

Example: Suppose the objective is to control a deterministic system. \( x_{t+1} = f(x_t, u_t) \in S_n \in U \), so as to minimize the cost \( \sum_{t=0}^T c(x_t, u_t) \) as a function of \( x_t \in S \). This objective function satisfies property 1. To see this, define
\[ A = S \times U, \quad H(a) = g(a) + L(f(a)) \]
and restrict the admissible policies to satisfy \( p_t(x_t) \leq x_t. \) If \( V \) is the infinitesimal operator, then (1) coincides with the so-called first-order minimization. Note that the only difference between the operator \( H \) (local income function in optimization problems) and the one used in previous abstract dynamic programming models (e.g., [2] and [5]) is that in these works, the mapping is defined on cartesian products of state and control spaces in lieu of action spaces. While both coincide in this example, the action space take on different roles in other problems in the sequel. As we know, we can indeed solve the problem in the present example using dynamic programming; however, this is because it is satisfied by other properties in addition to Property 1, which are characterized next.

**Definition.** A feasible valued function of states \( F^i : S_i \rightarrow Q \) is a cost-to-go function if
\[ F^i(x) = V_{\mu}(L_{\mu}(x)) \quad \forall x \in S, \]

\[ F^i(x) \] is defined for the noncooperative problem.
Theorem 2.1. Suppose that property B1 holds and define the following sequence of functions of states $B': S^* \to Q^*$, $i = 0, \ldots, N$.

(4a) $B^0(x) = V_N(L^0_N(x))$,

and

(4b) $B^{i+1}(x) = V_{i+1}(H' B^i(\mu_{i+1}(x)))$.

Then $B^i$, $i = 0, \ldots, N$ are cost-to-go functions if and only if

(5) $V_{i+1}(H' L^i_N(\mu_{i+1}(x))) = V_{i+1}(H' V_N(L^i_N(\mu_{i+1}(x)))$ for all $x \in S_{i+1}$, $i = 1, \ldots, N$.

Proof. Because of (4a) it suffices to show that, for each $i = 1, \ldots, N$, if $B^i = V_N(L^i_N)$, then $B^{i+1} = V_{i+1}(L^i_N)$ is equivalent to (5); but this follows immediately from (2) and (4a).

Corollary. Suppose that B1 and the following condition are satisfied:


(6) $V_{i+1}(H' L^i_N) = H' V_N(L^i_N)$, $i = 1, \ldots, N$.

Then, $B^i$, $i = 0, \ldots, N$ are cost-to-go functions and in particular $V_N(L^N_N) = B^0$.

Property B2 can be represented in the following commutativity diagram:

\[
\begin{array}{ccc}
Q^N \times M^N & \xrightarrow{H'} & Q^{N-1} \times M^{N-1} \\
V_N \downarrow & & \downarrow V_N \\
Q^N \times S^N & \xrightarrow{H} & Q^{N-1} \times S^{N-1}
\end{array}
\]

where the set of functions of states (resp. actions) parametrized by the elements of $M^N$ is denoted by $Q^N \times M^N$ (resp. $Q^{N-1} \times M^{N-1}$). The mappings $Q^N \times M^N \to S^N$, $Q^{N-1} \times M^{N-1} \to Q^{N-1}$ and $Q^{N-1} \times M^{N-1} \to Q^{N-1}$ induced by the pointwise application of the operators $V_N$ and $H'$ are denoted with the same symbol. In the special case of optimization problems, property B2 can be viewed as a formalization of a general optimality principle: in order to find the optimal partial return for each action $a$, it suffices to compute the cost-to-go function at the next stage and evaluate the local income function at $a$.

The fact that the recursions (2) and (4) are defined backwards is only due to the ordering of the operators $V$ in (1). More reversal of the stage indices results in a forward solution of $V_N(L^N_N(x))$, $x \in S_0$. The corresponding decomposability and commutativity assumptions are, in this case.

F1. Forward decomposability of the objective function.

(2) $L^i_N(\mu_{i+1}(x)) = H' L^i_N(\mu_{i+1}(x))$, $x \in S_{i+1}$, $i = 0, \ldots, N-1$.

where

$\begin{align*}
L^i_N: S^* \to Q^i, i = 0, \ldots, N, \\
H': Q^N \to Q^{N-1}, i = 0, \ldots, N-1.
\end{align*}$

F2. Forward commutativity of operators. 

(6) $V_i(H' L^i_N) = H' V_i(L^i_N)$, $i = 0, \ldots, N-1$.

Then we have

Theorem 2.2. Define the functions $F^i: S^* \to Q^i$, $i = 0, \ldots, N$ through the recursion:

(4a') $F^0(x) = V_0(L^0_N(x))$.
and

\[ F^{**}(x) = \mathbf{V}_{\mathbf{r}^c}(H^{**}(\mathbf{r}^c(x))) \]

Then, under assumptions F1 and F2, F' are cost-to-arive functions, i.e., F'(x) = \mathbf{V}_{\mathbf{r}^c}(\mathbf{L}^c_{\mathbf{r}^c}(x))

If either pair of assumptions is satisfied for a particular problem, then the other
one is trivially satisfied for the time-reversed problem. Hence, it is only meaningful to
distinguish between the forward and backward versions with respect to the state
evolution of the original problem. Rather than presenting only one of the versions
and fitting every particular example by possibly reversing the stage indices, we choose to
maintain always the original indices and present both the forward and the backward
formulations. This is due both to the fact that some problems are solved by forward
and backward recurrences concurrently (4.4), and because recurrences which evolve is
the direction of the system are of interest in some applications (e.g., Viterbi's forward
dynamic programming algorithm for real-time decision problems [8]).

When this general framework is applied to specific operators H and V, the
verification of the commutativity property (5), or the stronger version (6), often requires
an inductive proof which is common to most problems. Based on such an induction,
the next result provides a sufficient condition for B1 (analogously for F2) that entails
the verification of the commutativity of H with a single operator V.

Theorem 2.3. Suppose that condition B1 is satisfied, and define the function
\[ L^c_{\mathbf{r}^c}(x) = x, \] \[ L^c_{\mathbf{r}^c}(x) = x, \] for all \( i \leq j \leq N \) and all \( \mathbf{r}^c \in M_{\mathbf{r}^c} 

\[ (7a) \]

\[ (7b) \]

\[ (8) \]

\[ V_j(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_j(x)), \]

for all \( i \leq j \leq N \) and all \( \mathbf{r}^c \in M_{\mathbf{r}^c} \).

Then condition B2 is satisfied.

Proof. Since \( L^c_{\mathbf{r}^c} \) is a particular instance of \( (8) \) for \( i = j = N \), results in \( V_N(H^{**}_{\mathbf{r}^c} = H^{**}_{\mathbf{r}^c}(V_N(x))) \) (condition B2 for \( i = N \)). Now fix \( k \) and suppose that \( (4) \) is satisfied for \( k+1, \ldots, N \). We will show that under condition (3), we have \( V_k(H^{**}_{\mathbf{r}^c} = H^{**}_{\mathbf{r}^c}(V_k(x))) \). The proof will be divided into two stages:

(1) If \( V_{k+1}(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_{k+1}(x)) \) and condition (8) holds, then \( V_k(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_k(x)) \)

for all \( i \leq j \neq N \) and all \( \mathbf{r}^c \in M_{\mathbf{r}^c} \).

(2) If (8) holds and \( V_k(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_k(x)) \)

for all \( i \leq k \leq N \) and all \( \mathbf{r}^c \in M_{\mathbf{r}^c} \),

then \( V_k(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_k(x)) \).

Relationship (a) can be proved by induction: Let \( i = j - 1 \), then for all \( x \in S \) and \( \mu, \nu \in M \), we have

\[ V_j(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_j(x)) = H^{**}_{\mathbf{r}^c}(V_j(V_{j-1}(x))) \]

where the second through sixth equalities follow from (2), (4), (7a), (8), and (7b). Respectively. Now suppose that \( V_j(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_j(x)) \)

for all \( i \leq j-1 \), then, for all \( x \in S \) and \( \mu, \nu \in M \),

\[ V_j(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_j(V_{j-1}(x))) \]

where the second through sixth equalities follow from (2), (4), (7a), (8), and (7b).

Equation (9) follows directly from condition (b) implies that

\[ V_{j+1}(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_{j+1}(x)) \]

So, it suffices to show that

\[ V_j(H^{**}_{\mathbf{r}^c}(x)) = H^{**}_{\mathbf{r}^c}(V_j(x)) \]

but this readily follows from (7) and (8).

Three main differences between dynamic programming models can be

(i) Attention is not restricted to

(ii) The system is illustrated below by several applications to consider nonexclusion of

(iii) In the present model the \( \mathbf{r} \) function maps functions of states into

Besides its notational convenience, the action-value policy functions of

(ii) In optimization problems, the

(iii) In optimization problems, the

\[ \min_{\nu \in \mathbf{r} \times \mathbf{r}} \mathbf{g}(\mathbf{r}(x), \nu) + [\mathbf{g}(x), \nu] + \mathbf{g}(x) \times \mathbf{r} \times \mathbf{r} \]
where the second equation follows from (8). In order to prove (b) it is enough to show that

\[ V_N(H^iV^i_1\ldots V^i_N) = H^iV_N(L^i_1\ldots L^i_N) \]

for all \( \mu_{N-1} \in M_{N-1} \), and

\[ V_N(H^jV^j_1\ldots V^j_{N-1}V^j_{N}) = H^jV_N(L^j_1\ldots L^j_{N-1}) \]

for all \( \mu_{N-1} \in M_{N-1} \) and \( j = k, \ldots, N \).

Equation (9) follows directly from condition (8). To show (10), note that the assumption in (b) implies that

\[ V_{i-1}(L^i_{k-1}) = L^i_{k-1} \]

for all \( \mu_{k-1} \in M_{k-1} \).

So, it suffices to show that

\[ V_j(H^jV^j_{N-1}V^j_{N}) = H^jV_N(L^j_{N-1}) \]

and this readily follows from (7) and (8). □

Three main differences between the above framework and previous abstract dynamic programming models can be underlined, namely,

(i) Attention is not restricted to extremization operators: \( \varphi \in M \rightarrow \varphi(\mu) \) or \( \varphi_{s,x}(\mu) \). Although, of course, this is the most important, it is both useful and interesting from a conceptual viewpoint to consider nonextremization operators. Note also that since we do not require the extremization operators to coincide at each stage we can deal, for example, with extremization problems where inf and sup operators occur successively (e.g., dynamic two-person zero-sum games, where computational schemes based on dynamic programming principles are ubiquitous (cf. [9, [10])).

(ii) In the present model the generating operator of the objective recursive function maps functions of states into functions of actions (as in Dynkin and Yushkevich [11]), rather than into functions of states. In contrast to [11] the duality between states and actions is carried one step further by defining the admissible policies as mappings from the state space to the action space rather than to the underlying control space. The stochastic control formulation of [5, [12, Part I] is equivalent to the special case in which the action space is \( A \subset S \times U \), where \( U \) is a control space and the admissible policies \( \mu \in M \) are such that the image of each state belongs to its fiber, i.e.,

\[ p_{j}(\mu(x)) = x \]

for all \( x \in S \).

Besides its notational convenience, the versatility of the use of general action spaces and action-valued policy functions affords a nice parallelism (§ 1) between forward and backward problems.

(iii) In optimization problems, the commutativity conditions of the present model are weaker than the sufficient conditions imposed on the generating operator by previous models (typically, monotonicity and continuity). In addition, it appears that the strong commutativity condition of Theorem 2.3 (between \( V_N \) and \( H_i \) for \( i \) is not more difficult to check directly than previous conditions. Although it is natural to impose the monotonicity condition in order to satisfy commutativity with extremization operators, it is not necessary to do so. For example, consider the problem

\[ \min_{\mu \in M_{N-1}} g(x_0, x_0) + [x_0(x_1, x_2) + \cdots + g(x_{N-1}, x_N)] + \psi(x_N) \]

for \( x_0, x_1, \ldots, x_N \in \mathbb{R} \).
The generating operator $H^i(x_{i-1}, x_i) = \mu(x_{i-1}, x_i) + L^i(x_i)$ is not monotone in the function of states, but yet satisfies the above commutativity condition and, therefore, (11) admits a dynamic programming solution. In order to illustrate how previous conditions imply commutativity, consider the setting due to Bertsekas [5, 12, Part 1], which encompasses previous abstract dynamic programming models with real-valued return functions. The finite-horizon assumptions in [12, Prop. 6.1] imply the following conditions:

C1. $H^i$ is monotone and for every $i > 0$, there exists $\mu^* \in M^i$ such that:

$$H^i|_{\mu^*}^{\mu^*} \leq H^i|_{\mu^*}^{\mu^*} \leq \mu^* \in M^i$$

C2. There exists a sequence of policies $\mu^*_k \in M^i$, $k = 0, 1, \ldots$, such that:

$$L^{(0)}_{\mu^*_k} \Rightarrow \inf_{\mu \in M^i} L^{(0)}_{\mu} \text{ and } H^i|_{\mu^*_k}^{\mu^*_k} \Rightarrow \inf_{\mu \in M^i} H^i|_{\mu}^{\mu}$$

Then, it is straightforward to show that (8) follows from either condition.

In stochastic control problems the key to the existence of $\mu^*$ and $\mu^*_k$ with the above properties is the fact that the dependence of $L^{(0)}_{\mu^*_k}(x)$ on $\mu^*_k(x)$ is only through $\mu^*_k(x)$, therefore, for every $\delta > 0$ there exists a uniformly $\delta$-optimum policy $\mu^* \in A^i$ (not necessarily in $M^i$) such that for all $x \in S$:

$$L^{(0)}_{\mu^*}(x) = \inf_{\mu \in A^i} \left\{ L^{(0)}_{\mu}(x) \right\} \leq -1/\delta$$

In particular cases such $\mu^*$ can be shown to belong to $M^i$ and mid restrictions on the cost-per-stages guarantee that $C1$ and $C2$ are satisfied. For example, consider a controlled Markov process problem with additive cost-per-stages:

$$H^i(a, x) = L(a, x) + \int_{\mathbb{H}} L^i(w) P_{\alpha}(dw|a), \quad a \in A^i, \quad x \in S^i$$

where $S^i \times A^i \times \mathbb{H}$ and all admissible policies $\mu \in M^i$, satisfy $p_i(x, a, \Delta) = x$ for all $x \in S^i$, then it is easy to see that $L^{(0)}_{\mu^*}(x)$ is the expected value of the cost of using $\mu^* \in M^i$ when the initial state is $x \in S^i$. To show the existence of $\mu^* \in M^i$ satisfying (12), it is enough to assume that the state and control processes are Borel; $A^i, S_i, k = 0, 1, \ldots$, $N$ are analytic sets; the cost-per-stage $g_i: A^i \to R$, $k = 0, 1, \ldots$, $N$ and the terminal cost $r: S \to R$ are lower semi-analytic; the transition functions are Borel measurable and $M^i = \{\mu \in A^i \}$ such that $p_i(x, a, \Delta) = x$, $x \in S^i$ and $\mu$ is universally measurable (see [13], [12, Prop. 7.5]). Now, if $\gamma_k(x, L^{(0)}_{\mu^*}(x)) > -\infty$ for all $x \in S$, then for every $\epsilon > 0$ we can choose $\mu^* \in M^i$ such that $C1$ is satisfied. On the other hand, if there exists $\mu^* \in M^i$ such that $H^i|_{\mu^*}(x) < -\infty$ for all $a \in A^i$, then using (12) we can select $\mu^*$ such that $C2$ is satisfied. Other stochastic control problems such as these with worst-case, rather than average, objective function and multiplicative nonnegative, rather than additive, costs-per-stage can be shown to satisfy commutativity via conditions C1 and C2.

3. Application to classes of backward and forward problems. Once we have illustrated briefly the application of the backward dynamic programming framework and associated commutativity conditions to a class of stochastic control problems, in this section we show the application of the framework of §2 to classes of backward and forward problems. Because of the causality relation among the state, the utility of the forward formulation is restricted to open-loop problems. The main purpose of the first specific model to be presented (a deterministic optimum control problem) is to illustrate the symmetry achieved thanks to the versatility of the present applications of nonextensive includes the conventional finite case.

3.1. Additive-coast determinstic

additive-cost optimum control of $k = 0, \ldots, N$ into the framework formulation presented here all formulations, which are unique, does not permit the implementation of the recursive programming. For $t$ identification:

$$H^i$$

and

$$M_t = \{\mu \in A^i \}$$

Note that there is a bijection b admissible controls. On define clear that

$$\inf_{\mu \in M^i} L^{(0)}_{\mu^*}$$

In the forward case we define

$$H$$

and

$$M_t = \{\mu \in A^i \}$$

In this case, there is no one-to-one

and the set of admissible costs to controls, then here we have (deterministic systems), hence the subset of $S_i \times U, k = 0, \ldots, N$, if $H^i$ is defined through $H^i \approx \ldots$$

$$\inf_{\mu \in M^i} L^{(0)}_{\mu^*}$$

and in both cases the strong satisfied

3.2. Recursive computer system ($Q^i, \ldots$, where ($Q^i$)}
\[ x, s, L(x) \] is not monotone in the
mutativity conditions and, therefore,
order to illustrate how previous
thing due to Bertsekas [5], [12, Pan-
programming models with real-
opinions in [12, Prop. 6.1] imply the
there exists \( \mu^* \in M \) such that
\[ \left( \sum_{k=0}^{\infty} H^t \right) \text{inf}_{\mu \in M} L_{\mu}^*(x) = \varepsilon. \]
follows from either condition.
existence of \( \mu^* \) and \( [\mu^*_n] \) with the
of \( L_{\mu_n}^*(x) \) on \( \mu_n \) is only through
uniformly \( \delta \)-optimum policy \( \mu^* \in \mathcal{A}^* \)
\begin{align*}
V_0[L_{\mu_n}^*(x)] > -\infty, \\
V_0[H^t L_{\mu_n}^*(x)] = -\infty.
\end{align*}
that to \( M \) and mild restrictions on the
listified. For example, consider a con-
\[ \mu_n(x) = \mu_{n_x} \]
\[ \forall x \in S_{n+1} \]
\[ \mu_n \text{ satisfy } p_n(\mu_n(x)) = x, \text{ for all } x \in S, \]
\[ \mu \in \mathcal{M} \]
value of the cost of using \( \mu \) \in \mathcal{M} \),
tence of \( \mu \in M \) satisfying (12), it is
\begin{align*}
\forall x \in S, \quad \forall k = 0, \cdots, N \quad \text{and the terminal cost}
\end{align*}
\[ \max \{ 0, x \} \text{ and then for every } \varepsilon > 0 \text{ we have}
\[ \forall x \in S, \quad \forall k = 0, \cdots, N \quad \text{we nonnegative, rather than additive,}
\end{align*}
\[ \text{the worst-case, rather than additive,}
\end{align*}
\[ \text{via conditions C1-C2.} \]
\begin{align*}
\text{Retrograde problems. Once we have illus-
\text{stochastic control problems, in this}
\text{pk of § 2 to classes of backward and}
\text{on among the state spaces, the utility-
\text{loop problems. The main purpose of
\text{the simultaneous optimum control problem is
\[ \text{to illustrate the symmetry achievable between the backward and forward formulations thanks to the versatility of the action space and action-valued polynomials. In § 3.2 we present applications of nonextremization operators in a general algebraic setting which includes the conventional finite state-space dynamic programming models.} \]
\[ \text{3.1. Additive-cost deterministic optimum control. It is easy to fit the problem of}
\text{additive-cost optimum control of a deterministic system} \quad x_{n+1} = f(x_n, u_n) \in S_{n+1}, \quad u_n \in U_n, \\
k = 0, \cdots, N \]
\[ \text{into the framework of § 2. It is noteworthy that the dynamic programming}
\text{formulation presented here allows a duality between the forward and backward}
\text{formulations that unlike previous works, [14, [15] which require the system to be}
\text{deterministic, does not impose any restrictions to define the forward dynamic}
\text{programming recursion. For the backward formulation we make the following}
\text{identifications:}
\begin{align*}
\text{A}_n = S_n \times U_n, \\
\text{H}^t L(a) = g(a) + L(f(a))
\end{align*}
\[ \text{and}
\begin{align*}
\text{M}_n = \{ \mu \in \mathcal{A}_n^* \text{ such that } p_n(\mu(x)) = x \text{ for all } x \in S \}.
\end{align*}
\[ \text{Note that there is a bijection between the set of admissible policies and the set of}
\text{admissible controls. On defining} \quad L_n \quad \text{through the recursion (2)—with} \\
\text{L}^\infty = 0 \text{—it is clear that}
\begin{align*}
\text{inf}_{\mu_n \in \mathcal{M}_n} L_{\mu_n}^*(x) & = \text{inf}_{\mu_n \in \mathcal{M}_n} \sum_{n=0}^{N} g_n(x_n, u_n), \\
\text{in the forward case we define}
\begin{align*}
\text{A}_n = S_n \times U_{n+1}, \\
\text{H}^t L(a) = g(a) + L(f(a))
\end{align*}
\[ \text{and}
\begin{align*}
\text{M}_n = \{ \mu \in \mathcal{A}_n^* \text{ such that } f_n(\mu(x)) = x \text{ for all } x \in S \}.
\end{align*}
\[ \text{In this case, there is no one-to-one mapping between the set of admissible policies}
\text{and the set of admissible controls (had we defined policies as mappings from states}
\text{to controls, then here we would be able to define the forward recursion only for}
\text{deterministic systems); however, there is indeed a bijection between} \quad M_{n+1} \times S_{n+1} \\
\text{and the subset of} \quad S_n \times U_n \times S_{n+1} \text{that represents the trajectories of the system. Hence,}
\text{if} \quad L \text{ is defined through}\ (3')\ —\text{with} \quad L^\infty = 0 \text{ then it can be checked that}
\begin{align*}
\text{inf}_{\mu_n \in \mathcal{M}_n} L_{\mu_n}^*(x) & = \text{inf}_{\mu_n \in \mathcal{M}_n} \sum_{n=0}^{N} g_n(x_n, u_n), \\
\text{and in both cases the strong commutativity property of Theorem 2.3 is obviously}
\text{satisfied.} \]
\[ \text{3.2. Recursive computation of stage-separated functions. Consider an algebraic}
\text{system} \quad (Q, +, \cdot), \text{ where} \quad (Q, +) \text{ is a commutative semigroup and the internal binary}
operation is left-distributive with respect to additions. Suppose that $S_i, i = 0, \ldots, N+1$ are finite sets, $s : S_i \times S_{i+1} \to Q$, and the goal is to compute

\[
\mathcal{L}_0(x) = \sum_{s \in S_0} \sum_{x_1 \in S_1} \cdots \sum_{x_N \in S_N} \mathcal{g}(x_1, x_2) \cdot \mathcal{g}(x_1, x_2) \cdot \mathcal{g}(x_N, x_{N+1})\]

for every $x \in S_N$. Some examples of problems of this type are:

(i) Shortest path is a shortest network with $(\mathbb{R}, \min)$.

(ii) Satisfiability of Boolean clauses with $(\{0, 1\}, \lor, \land)$.

(iii) Computation of marginal probability distributions, with $(\mathbb{R}^+, +, \cdot)$ (§4).

(iv) Symbolic transfer function problems, with $S \times \mathbb{S}$, where $S$ is the family of sets of finite-length strings of symbols drawn from a finite alphabet, and $\mathbb{S}$ denotes string concatenation (see [16] for a generalization of the McNaughton-Yamada algorithms [17] for the computation of regular expressions given arbitrary state graphs).

(v) Dynamic programming for optimization problems where the return space is only partially ordered. If $(Q, \leq, \triangledown)$ is a conditionally complete associative lattice [7], then the algebraic system $(\mathbb{R}^+, \max, \cdot)$ satisfies the above properties, where $\max (A, B)$ is the set of maximal elements of $A \cup B$ and $\cdot$ is only defined $(a\cdot b=(a+b, a \circ b))$ when the left operand is a singleton. Interestingly, the generalized version of the optimality principle given by Brown and Strauch [7] (see also [18]) is a special case of the commutativity condition (1).

The function in (13) can be computed by a backward dynamic programming recursion. Define

\[
A_i = S_i \times S_{i+1},
\]

\[
V_i(q \mu) = \sum_{\mu \in A_i} q(\mu),
\]

\[
H_i(\alpha) = g_i(\alpha) \cdot L_i(\rho)(\alpha),
\]

\[
M_i = \mu \in A_i \exists x^* \in S_{i+1}, \text{such that } \mu(x) = (x, x^*) \text{ for all } x \in S_i.
\]

If the recursion (2) is used to define $L_0^\ast$ with $L_0^\ast(x) = g_0(\mu(x))$ then it is easy to see that $L_i(x) = V_i(x, L_i^\ast(x))$. The commutativity condition of Theorem 2.3 follows from the left-distributivity of $\cdot$ with respect to $\circ$. Analogously, if we identify $A_i = S_i \times S_{i+1}, M_i = \mu \in A_i$; there exists $x^* \in S_{i+1}$, such that $\mu(x) = (x, x^*)$ for all $x \in S_i$ and $H_i(\alpha) = L_i(\rho)(\alpha)$, then recursion (5) can be used to define $L_i^\ast$ with

\[
L_i^\ast(x) = g_i(\mu(x)),
\]

and right-distributivity of $\cdot$ with respect to $\circ$ implies that forward dynamic programming can be used to recursively compute

\[
\sum_{x \in S_i} \cdots \sum_{x_1 \in S_1} \cdots \cdots \sum_{x_N \in S_N} \mathcal{g}(x_1, x_2) \cdot \mathcal{g}(x_1, x_2) \cdots \cdot \mathcal{g}(x_N, x_N)
\]

for all $x \in S_{N+1}$.

Note that if $(Q, +)$ has an identity element 0 which is also an annihilator of $\cdot$, i.e., $0 + x = x + 0 = 0$ for all $x \in Q$, then the special case of the above formulation is

\[
\sum_{x \in S_i} \cdots \sum_{x_1 \in S_1} \cdots \cdots \sum_{x_N \in S_N} \mathcal{h}(x_1, u_1) \cdot \mathcal{h}(x_1, u_2) \cdots \cdot \mathcal{h}(x_N, u_N) \cdot \mathcal{g}(x_N, x_{N+1})
\]

subject to $x_{N+1} = f(x, u_1, u_2, \ldots)$.

To see how, let

\[
T_i(x, u) = \{u \in I_i \text{ such that } f(x, u_i) = x_{N+1}\}
\]

and use the associative and distributive properties of $\cdot$ and $+$.

4. Backward-forward finite-h

4.1. General setting and some operators $V : Q^0 \to Q^0, i = 1, \ldots, N$, is to find $V_i(x_1, \ldots, x_N)(d_1, \ldots, d_N)$, that these sets, operators and functions $B_i$, there exists backward properties $\mathcal{B}_1$ and $\mathcal{B}_N$, $\mathcal{B}_i \subseteq Q^0, i = 1, \ldots, N$, such that $d_i \in B_i$, for all $d_i \in B_i$ and $x_N$.

By $V_i : Q^{0, \infty}$ (e.g., [19]): the open generating general commutativity condition $Q^0$. Let $x \in \mathbb{N}$ and $x_N$.

Fix an index $1 \leq i \leq N$ and $N$, we can solve $V_{i+1}(V_i(x_1, \ldots, x_N), x_N)$ forward dynamic programming $g$ for each index and each element forward search $V_{i+1}(V_i(x_1, \ldots, x_N), x_N)$ for all elements and stages is given.

THEOREM 4.1. Suppose that $V_i(V_i, V_i, \leq) \in \mathbb{N}$. Define the function

\[
B_i^{-1}(x) = V_i^{-1}(V_i^{-1}(V_i(x)), 1)
\]

\[
F_i^{-1}(x) = V_i^{-1}(V_i(x)), 1)
\]

Then there exist functions $W_i$.

\[
V_i(x_1, \ldots, x_n)(d_1, \ldots, d_n)
\]

Proof. The quantity $V_{i+1}(V_i, x_1, \ldots, x_N, d_N)$, and $N$, we have $V_{i+1}(V_i)(x_1, \ldots, x_N)$.

Now, it will be shown that for an only through $B_i$, and hence the

\[
\begin{align*}
\text{and} \\
g(x_1, x_2) &= \begin{cases}
1 & x_1 = x_2 \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]
of this type are:

\[ \sum_{x \in A(x_0)} \mu(x) \text{ if } a \in A_{x_0} \]

\[ \sum_{x \in A(x_0)} \mu(x) \text{ if } a \notin A_{x_0} \]

and use the associative and distributive properties of the algebraic system.


4.1. General setting and sufficient conditions. Given a collection of sets \( D_i \) of operators \( V_i, Q_i Q_i^2 Q_i \) for \( i = 1, \ldots, N \) and a function \( f: D_i \times Q_1 \rightarrow Q_1 \) suppose that the goal is to find V_{i+1}, V_{i+2}, \ldots, V_N, f for all \( i \in D_i \) and \( i = 1, \ldots, N \). Assume that these sets, operators and function \( f \) satisfy the following conditions:

BF1. There exists a forward and backward decomposable functions (i.e., satisfying properties B1 and F1, respectively) \( f_{B1}(\mu_{i+1}, \mu_i) \), \( f_{F1}(\mu_{i+1}) \), \( f_{B1}(\mu_i) \), \( f_{F1}(\mu_i) \) and \( f_{B1}(\mu_{i+1}, \mu_i) \), \( f_{F1}(\mu_{i+1}) \), \( f_{B1}(\mu_i) \), \( f_{F1}(\mu_i) \) that satisfy the strong commutativity condition (8).

BF2. Denote by \( \Psi_i : Q_i \times Q_i \rightarrow Q_i \) the mapping induced by \( f_i \), i.e., \( \Psi_i(f_i) = f_i \). The operators \( V_i : \Psi_i : Q_i \times Q_i \rightarrow Q_i \) are bounded, and \( V_i : \Psi_i : Q_i \times Q_i \rightarrow Q_i \) is the generating operator of the backward function in BF satisfy the strong commutativity condition (8). (Analogously, with \( V_i : \Psi_i : Q_i \times Q_i \rightarrow Q_i \) is solved simultaneously for all elements and stages is given by the following result.

Theorem 4.1. Suppose that conditions BF1 and BF2 are satisfied and that \( V_N = V_{N+1} \). Let \( f^{(n)}(x) \) and \( f^{(n)}(x) \) be the following functions:

\[ f^{(n)}(x) = \sum_{x' \in A(x)} \mu(x') f_i(x', x') \]

\[ f^{(n)}(x) = \sum_{x' \in A(x)} \mu(x') f_i(x', x') \]

Then there exist functions \( W_i : Q_i \times Q_i \rightarrow Q_i \), \( i = 1, \ldots, N \) such that

\[ W_i(\mu_{i+1}, \mu_i) = \sum_{x' \in A(x)} \mu(x') f_i(x', x') \]

\[ W_i(\mu_{i+1}, \mu_i) = \sum_{x' \in A(x)} \mu(x') f_i(x', x') \]

Proof. By the definition of \( V_{i+1}, V_{i+2}, \ldots, V_N, f \), f is a function of \( \mu_i, \mu_{i+1}, \mu_{i+2}, \ldots, \mu_{i+N} \) for \( i = 1, \ldots, N \). Theorem 4.1 will follow by showing that the dependence of \( V_{i+1}, V_{i+2}, \ldots, V_N, f \) on \( \mu_i, \mu_{i+1}, \mu_{i+2}, \ldots, \mu_{i+N} \) is only through \( \Psi_i, f_i \), and hence the same is true for the right-hand side of (16). Using the
in Theorem 2.3, we have (note that here the horizon for the backward problem is N = 1) $P_{n \rightarrow m}(x_0) = \mathbb{E}_{n \rightarrow m}(x_0)$.

Furthermore, the result in part (a) of the proof of Theorem 2.4 (the conditions for its validity are guaranteed by properties B1 and B2) implies that

$$\mathbb{E}_{n \rightarrow m}(x_0) = \mathbb{E}_{n \rightarrow m}(x_0) = \cdots = \mathbb{E}_{n \rightarrow m}(x_0).$$

But according to (7b), the right-hand side of (17) depends on $M_{n \rightarrow m}$ only through the function $F_{n \rightarrow m}$, which because of (7a) can be written as $F_{n \rightarrow m}(x) = \mathbb{E}_{n \rightarrow m}(x)$

$$= \mathbb{E}_{n \rightarrow m}(F_{n \rightarrow m}(x)) = \mathbb{E}_{n \rightarrow m}(F_{n \rightarrow m}(x)) = \mathbb{E}_{n \rightarrow m}(F_{n \rightarrow m}(x)),$$

where the second equation follows from (2), and the third and fourth equations follow from property B2 (which is satisfied because of the strong commutativity condition (3)) and the corollary to Theorem 2.1, respectively.

Using the fact that $V_{n \rightarrow m} = V_{n \rightarrow m}$, we can write

$$V_{n \rightarrow m}(x, a) = \mathbb{E}_{n \rightarrow m}(x, a) = \mathbb{E}_{n \rightarrow m}(x, a) = \mathbb{E}_{n \rightarrow m}(x, a).$$

and hence, by analogy, it follows that the dependence of $V_{n \rightarrow m}(x, a)$ on $D_{n \rightarrow m}$(1) is through the cost-to-arrive function $F_{n \rightarrow m}$.

From the above proof it is easy to see that the conditions of Theorem 4.1 guarantee the validity of a backward-forward recursion for problems where several consecutive immediate elements are fixed, i.e. (15) can be generalized to

$$V_{n \rightarrow m}(x, a) = \mathbb{E}_{n \rightarrow m}(x, a) = \mathbb{E}_{n \rightarrow m}(x, a) = \mathbb{E}_{n \rightarrow m}(x, a).$$

Perhaps the simplest example of the backward-forward model is the problem mentioned in the introduction: given a layered network, find for each arc in the network the shortest path from source to destination that contains that arc. The straightforward approach is to run a forward or backward iteration for each arc (deleting all other arcs in the same layer); however, even if we take advantage of the obvious commonality of some of the computations, the number of steps required by this approach is quadratic in the number of layers. In contrast, if the result of Theorem 4.1 is employed (note that the minimization operators commute), then the solution to the shortest path problem requires only two independent (one forward and one backward) dynamic programming recursions, which simply compute the cost of each node from the source to and from the destination. Once the cost-to-arrive and cost-to-go are computed for all nodes in the network, the solution is given by $W_{n \rightarrow m}(F_{n \rightarrow m})$ which is simply the sum of the length of each arc and the distances of its head and tail to the destination and from the source, respectively. The next subsection illustrates the application of the backward-forward dynamic programming setting to the problem of finding the sequence of joint distributions of consecutive states of a discrete-time Markov process and the problems of fixed-interval estimation and smoothing.

4.2. Applications. Consider a Markov process $(X, (Q, F)) = (Q_0, F_1, F_2, \ldots, N)$ whose finite-dimensional distributions are determined by $P$, an arbitrary

\begin{equation}
\begin{aligned}
\text{initial probability measure on } (Q, F), \hspace{1cm}
p(H) &= n \in \mathbb{F}^a_n \text{ is measurable for each } B \in \mathcal{F}, \hspace{1cm}
\text{on joint distributions of consecutive states,}
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{and for } i = t + 1 \text{ is also possible,}
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{and hence, the sought-after relationship is}
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{next we examine another }
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{with non-anticipation properties of probability-of-error detection.}
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{the sub-algebra generated by the sequence of transmitted symbols of}
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{finite sets. Optimum decisions bo}
\end{aligned}
\end{equation}
initial probability measure on \((\Omega_i, F_i)\), and by the transition functions \(P_i, i = 1, \ldots, N\) such that \(P(x, \cdot)\) is a probability measure on \((\Omega_i, F_i)\), for each \(x \in \Omega_i\), and \(P_i(\cdot, B)\) is measurable for each \(B \in F_i\). Suppose that the objective is to obtain the sequence of joint distributions of consecutive states, namely,

\[
P(X_{i} \in B_i, X_{i+1} \in B_{i+1}) = \sum_{x_i \in \Omega_i} \sum_{x_{i+1} \in \Omega_{i+1}} P_i(dx_{i+1}) \prod_{i=1}^{N} P_i(dx_i, dw_i).
\]

In order to put this problem in the backward-forward framework, let \(D_i = \Omega_i \times F_i\), 
\(V_i(q(d_i)) = q_i(f_i(x_{i-1}, w_i))\) for \(q_i(x, B) = \sum_{x_i \in \Omega_i} \sum_{x_{i+1} \in \Omega_{i+1}} P_i(dx_{i+1}) \prod_{i=1}^{N} P_i(dx_i, dw_i)\). We make the following identifications for the state, action, policies and generating operators of the backward and forward formulations:

\[
\begin{align*}
\bar{S}_i &= \Omega_i, \\
\bar{A}_i &= \Omega_i \times D_{i+1}, \\
\bar{S}_{i+1} &= F_i, \\
\bar{A}_{i+1} &= D_{i+1} \times \Omega_i, \\
\bar{R}(s) &= P_i(p_i(a), p_i(a)|L(p_i(a), \cdot)), \\
\bar{H}(s) &= P_i(p_i(a), p_i(a)|L(p_i(a), \cdot)), \\
\bar{M}_{i} &= (\mu \in \bar{A}_i^2), \text{ there exists } (w, B) \in \Omega_i \times F_{i+1}, \\
\bar{M}_{i+1} &= (\mu \in \bar{A}_{i+1}^2), \text{ such that } \mu(s) = (x, w, B) \text{ for all } s \in \bar{S}_i, \\
\bar{M}_{i+1} &= (\mu \in \bar{A}_{i+1}^2), \text{ such that } \mu(E) = (w, R, E), \text{ for all } E \in \bar{S}_i.
\end{align*}
\]

Note that the cost-to-arrive function is now a probability measure and that there is a one-to-one correspondence between \(D_i\) and \(\bar{M}_{i+1}\), and between \(D_i\) and \(\bar{M}_i\). The commutativity conditions follow in this case from the linearity of the integral, and the problem can be solved by either a forward or a backward recursion with respective value functions:

\[
F^{j+1}(E) = \int_{\bar{S}_j} P_{j+1}(x, E) F^{j}(dx_j), \quad F^{1}(E) = P_1^{j}(E)
\]

and

\[
B^{j+1}(s) = \int_{\bar{S}_j} P_{j+1}(x, dw_i) B^{j}(dx_j), \quad B^{N+1} = 1.
\]

Moreover, because of Fubini's theorem, a backward-forward solution given by (1b) with \(j = 0\) is also possible:

\[
V_{X_1, \bar{X}_1} = \left[ F^{1}(C_{1}) P_{2} \left( x_1, C_{2} \right) B^{2}(x_1) \right] + \text{setting to the problem of finding the states of a discrete-time Markov process, and smoothing.}
\]

process \([X_n, (\Omega, F) = (\Omega_i, F_i), \bar{X}_n] \text{ are determined by } P_i, \text{ an arbitrary}


can be made according to various optimality criteria; for example, the receiver may select the sequence in $U_{0,N}$ that maximizes $P^2[u_0, \ldots, u_N]$ (maximum likelihood sequence detection), or the sequence of arguments that maximizes the marginal, $P^2[u_i], i = 0, \ldots, N$ (minimum error probability detection). In data-transmission problems such as asynchronous demultiplexers, transmission of convolutionally encoded data and intersymbol interference problems, the a posteriori distribution can be decomposed in product form:

$$P^2[u_0, \ldots, u_N] = \prod_{x_0} A_0(x_0, u_0) \ldots \prod_{x_N} A_N(x_N, u_N)$$

where $x_0 = x(u_0, u_0)$ and $x_N$ is $G$-measurable.

The maximization of the joint distribution (maximum likelihood sequence detection) is a deterministic optimum control problem which fits into the framework presented in §3.2, and hence can be solved by either a backward or a forward recursion (in real-time applications, the latter is employed in a near-optimum version where decisions are made after a fixed lag—the Viterbi algorithm [9]). If, instead, the optimality criterion is minimum probability-of-error, then the central task of the detector is to compute the marginal a posteriori distribution of each transmitted symbol, i.e.,

$$P^2[u_i] = \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_{N}} \prod_{k=0} A_k(x_k, u_k), \quad i = 0, \ldots, N$$

This problem fits also in the framework of §3.3 and can be solved also by backward or forward dynamic programming. The forward recursion is simplified by noting that in the foregoing data-transmission problems the following condition holds:

St: For $k = 0, \ldots, N$, if there exists $x \in \Omega_k$, $u \in U_k$ and $u' \in U_k$ such that $f_k(x, u) = f_k(x, u')$ then $u = u'$.

Then the corresponding value functions are as follows:

$$V^2_k(x) = \sum_{x' \in \Omega_{k-1}} B^2(x', f_k(x, u_0))A_k(x', u)$$

and

$$B^2(x) = \sum_{x' \in \Omega_{k-1}} B^2(x', f_k(x, u_0))A_k(x', u_0).$$

In the problem of intersymbol interference a forward dynamic programming solution to the problem of computing the marginal distributions has been reported by Hayes, Cover and Rote [20]. The main shortcoming of this algorithm is that it requires a separate recursion for each value of each transmitted symbol. More efficient solutions is possible by realizing that (21) fits the backward-forward framework of this section, because it can be solved by either backward or forward recurrences and its operation (summations) commute. It is easy to check that in this case (15) takes the form

$$P^2[u_i] = \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_{N}} \prod_{k=0} A_k(x_k, u_k) \sum_{x_0} F^2(x_0, x_i, u_0)B^2(x_i)B^2(x_i)$$

which further reduces to

$$P^2[u_i] = \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_{N}} F^2(x_i, x_i, B^2(x_i))$$

if the following condition is satisfied.

32. For $k = 0, \ldots, N$, if there exist $f_k(x, u) = f_k(x', u')$ then $u = u'$.

Then, as in the problem of fit backward-forward solution of the fusion problem, exhibits linear compi-

tract to the quadratic complexity. Another illustration of the applicability of the problem of fixed-interval maximum time Markov process, i.e., find

$$\arg \max_{u \in U_k} \max_{x' \in \Omega_{k-1}} p(x', u)$$

assuming that conditional prob

$$p(x', u) = p(x', u|x)$$

unconditional density and the variate then we have

$$\arg \max_{u \in U_k} \max_{x' \in \Omega_{k-1}} p(x', u|x)$$

where

$$F_{t_{k-1}}(x_{t-k}) = \arg \max_{u \in U_k} p(x_{t-k}, u|x)$$

and

$$R_{t_{k-1}}(x_{t-k}) = \max_{u \in U_k} p(x_{t-k}, |x)$$

Then, the optimization in (23) can be recursion because

$$\max_{u \in U_k} \max_{x' \in \Omega_{k-1}} p(x', u|x)$$

(29)

$$\max_{u \in U_k} \max_{x' \in \Omega_{k-1}} p(x', u|x)$$

Once the optimum terminal state is $x_{t-1}$ in (26) can be recovered by backtraceforward or backward recurrences. The recurrence, in which one compares $t_{k-1}$ in (27) and (28), respectively, and then

$$\arg \max_{u \in U_k} \max_{x' \in \Omega_{k-1}} p(x', u|x)$$

(30)
if the following condition is satisfied (e.g., frequently \( f_k(\cdot, \cdot), k = 0, \ldots, N \) is a shift register system).

S2. For \( k = 0, \ldots, N \), if there exists \( x, x' \in \Omega, u \in U, u' \in U \) such that \( f_k(x, u) = f_k(x', u) \) then \( u = u' \).

Thus, as in the proof of finding the shortest path through every arc, the backward-forward solution of the fixed-interval minimum probability-of-error detection problem exhibits lower complexity in the number of transmitted symbols in contrast to the quadratic complexity of the Haver-Cover-Kaiser algorithms [20].

Another illustration of the applicability of the backward-forward framework is the problem of fixed-interval maximum a posteriori sequence smoothing of a discrete-time Markov process, i.e.,

\[
\arg \max_{x_{n+1}^N} \arg \max_{x_0^N} p(x_0^N, x_{n+1}^N | y_0^N, y_{n+1}^N)
\]

assuming that conditional probability density functions exist and that \( p(y_0^N, y_{n+1}^N | x_0^N, x_{n+1}^N) = \Pi_{k=0}^{N-1} p_k(x_k | x_{k-1}) \). If \( p_k(x_k) \) and \( q_k(x_k | x_{k-1}) \) denote the unconditional density and the transition density of the Markov process respectively, then we have

\[
\arg \max_{x_{n+1}^N} \arg \max_{x_0^N} p(x_0^N, x_{n+1}^N | y_0^N, y_{n+1}^N) = \arg \max_{x_0^N} \max_{x_{n+1}^N} p(x_0^N) \prod_{k=0}^{N-1} q_k(x_k | x_{k-1}) p_k(x_k | x_{k-1}).
\]

Identifying \( S_0 = \Omega, A_0 = \Omega, A_{k-1} = \Omega, A_k = A_{k-1} \) and \( M_0 = [u : \text{there exists } x^* \in S_{\infty}, u(x) = (x^*, x)] \) and \( M_1 = [u : \text{there exists } x^* \in S_{\infty}, u(x) = (x, x^*)] \), we define the recurrences:

\[
F_0(x_0) = p_0(x_0) \prod_{k=0}^{N-1} q_k(x_k | x_{k-1}) p_k(x_k | x_{k-1}),
\]

and

\[
B_0 = 1,
\]

\[
R_i(x_{n+i}) = \max_{x_{n+i}} p_i(x_{n+i} | x_n) p_{n+i}(x_{n+i} | x_{n+i-1}), \quad i = 1, \ldots, N.
\]

Then, the optimization in (26) can be carried out by either a backward or a forward recursion because

\[
\arg \max_{x_{n+1}^N} \arg \max_{x_0^N} p(x_0^N, x_{n+1}^N | y_0^N, y_{n+1}^N) = \arg \max_{x_0^N} F_0(x_0) = \arg \max_{x_n} p_n(x_n) p_{n+1}(x_{n+1} | x_n) B_n(x_n).
\]

Once the optimum terminal state is obtained through (29), the maximizing sequences in (26) can be recovered by backtracking the optimum transitions resulting from the forward or backward recursion. The alternative to this method is the backward-forward recursion, in which one computes both the out-to-arrive and the out-to-go through (27) and (28), respectively, and then solves for (cf. Theorem 4.1)

\[
\arg \max_{x_{n+1}^N} \arg \max_{x_0^N} \arg \max_{x_0^N} p(x_0^N, x_{n+1}^N | y_0^N, y_{n+1}^N)
\]

\[
\arg \max_{x_{n+1}^N} \arg \max_{x_0^N} F(x_n) B_n(x_n), \quad i = 0, \ldots, N.
\]
ABSTRACT

It can be shown that (37) holds if $L^{\kappa}$, $x \equiv N = 1, 2, \ldots$ (i.e., commutative condition is satisfied). 

$$\lim_{n \to \infty} V_{\infty}(L^{\kappa} (x)) = x \equiv N \quad \text{s.t.} \quad x \geq 0, 1, \ldots$$

As illustrated by the following notvarial. Consider the algebraic $x$.

$$\mathcal{A}(x, k) = \left\{ \begin{array}{ll}
0 & x \equiv 1, 2, \ldots \\
1 & x \equiv 0, 1, \ldots
\end{array} \right.$$

In this case any infinite sequence of $x \equiv k$ then $g(x, k) x \equiv 1, 2, \ldots$ such that $V_{\infty}(L^{\kappa} (x)) = x \equiv 0, 1, \ldots$.

The second question of interest whether the sought-after function $V$ problems (i.e., $S_n$, $A$, $H^n$, $V_j$, $f_j$ for $x$ is obviously the case, because in that it does not depend on the stage-in for a particular problem the following be a fixed point of $T$ in the station).

B4. $\mathcal{H}(V_{\infty}(L^{\kappa} (x)) at a$ all of $A$.

Note that the above counterexample: contraction assumptions of Denare linearities conditions of Bertsekas[5] and B4 are satisfied in stationary problems, another question of interest obviously this problem has no co operators $V$.

5. A glimpse at infinite-horizon models. The finite-horizon commutativity conditions of $\mathcal{B}$ are not sufficient to ensure the validity of the dynamic programming recursion in infinite-horizon operator models. In our present section this problem in the general nonstationary case for backward models. Forward counterparts of all results can be obtained following the approach of § 2. Furthermore, the fixed-point property of the sought-after function of states is studied in the stationary case.

Suppose that sequences of operators $\{V_i, Q_i, T_i, A_i, \ldots \}$, $\{H^i, Q^i, \}$, $i = 1, 2, \ldots$ and a sequence of functions of states $f_j : S_j \to Q$, $i = 1, 2, \ldots, j \in \mathbb{Y}$ is given. Define $L^{\kappa} \equiv A \in \mathbb{Y} \times (V)$ and $L^{\kappa}(x) = \mathcal{V}(\mathcal{B}^x (x))$. The first goal is to impose conditions to guarantee that the function $V_{\infty}(L^{\kappa} (x)) = x \equiv 0, 1, \ldots$ can be obtained by a (backward) infinite-horizon dynamic programming recursion, or more generally that $\lim_{n \to \infty} T^{\kappa} L^{\kappa} = T^{\kappa} L^{\kappa}$ is a cost-to-go function, i.e.,

$$\lim_{n \to \infty} T^{\kappa} L^{\kappa} = \lim_{n \to \infty} T^{\kappa} L^{\kappa}$$. 

where the operator $T_1, Q^{\kappa} \equiv 0$ is defined by

$$T_1(x) = \mathcal{V}(H^{\kappa} (L^{\kappa}(x)))$$. 

$^1$K denotes a generic state-independent term which need not coincide in different equations. 

References:

3. G. L. NEUMANN, Introduction to Di.
$\mu$ can be shown that (37) holds if the cummutativity condition B2 is satisfied for $L_{\mu}^{\alpha}$, $\alpha = 1, 2, \ldots$ (i.e., commutativity holds for finite-horizons) and the following condition is satisfied.

B3. \( \lim_{n \to \infty} V_{n}(L_{\mu}^{\alpha}(x)) \) and \( V_{n}(\lim_{n \to \infty} L_{\mu}^{\alpha}(x)) \) exist and are equal for all $x \in S$, and $0 \leq i \leq 1, \ldots$.

As illustrated by the following example, the equality of the functions in B3 is nottrivial. Consider the algebraic system (see §2.2) \((Q, \cdot, \cdot) = ([0, 1], OR, AND), \) let $x \in [0, 1, \ldots]$ and

\[ g(x, k, j) = \begin{cases} 1, & \text{if } k \neq 0 \text{ and } j = 0 \text{ or } k + 1, \\ 0, & \text{otherwise}. \end{cases} \]

In this case any infinite sequence of states \((x_k)\) results in $\lim_{n \to \infty} V_{n}(x_k, k_n) = 0$; however, if $k \neq N$ then

\[ g(x_k, k, k-1) \cdot g(x_k, k + 1, k+1) \cdot g(x_k, k-N, k-N) = 1. \]

Therefore we have that $V_{n}(\lim_{n \to \infty} L_{\mu}^{\alpha}(j)) = 0$ for $j = 0, 1, \ldots$, while $lim_{n \to \infty} V_{n}(\{0\}) = 1$.

The second question of interest in connection with infinite-horizon models is whether the so-called after function $V_{\infty}(\lim_{n \to \infty} L_{\mu}^{\alpha}(j))$ is a fixed point of $T$ in stationary problems (i.e., $S_\infty, A_\infty, H, V_{\infty}, T$ do not depend on the stage-index $\alpha$). If (18) holds this obviously is the case, because in the stationary case if $V_{\infty}(\lim_{n \to \infty} L_{\mu}^{\alpha}(j))$ exists then it does not depend on the stage-index. Nevertheless, even if $B2$ or $B3$ fail to be true for a particular problem the following condition is sufficient for $V_{\infty}(\lim_{n \to \infty} L_{\mu}^{\alpha}(j))$ to be a fixed point of $T$ in the stationary case.

B4. $V_{\infty}(\lim_{n \to \infty} L_{\mu}^{\alpha}(a))$ and $V_{\infty}(\lim_{n \to \infty} HU_{\infty}^{\alpha}(a)))$ exist and coincide for all $a \in A$.

(Note that the above counterexample to B3 satisfies B4.) It can be checked that the contraction assumptions of Denardo [2] and the continuity, uniform growth and linearity conditions of Bertsekas [5] along with the monotonicity of $H$ imply that $B3$ and $B4$ are satisfied in stationary information problems. In connection with these problems, another question of interest is the existence of $(r)$ optimal stationary policies; obviously this problem has no counterpart in our formulation with more general operators $V$.

REFERENCES


1. Introduction. Recently much effort has been made to systems and periodic optimization problems. It is obvious for this reason that there are some difficulties. But another important aspect is that the system is subject to time-varying disturbances. In [10] one of the problems for a periodic system in the presence of these disturbances is a Riccati equation for stochastic differential equations with periodic coefficients. The separation principle holds for periodic functions. Periodic functions are easy to handle. As we can see from simple examples, periodic functions are often periodic, but almost periodic functions can be considered as a special case of periodic functions.

2. The semigroup model. In this section, we consider the periodic perturbation in [19].

2.1. Almost periodic solutions of a Hilbert space with inner product b(y) function in Y. It is said to be almost periodic if there is a sequence a_n, such that lim n→∞ a_n