

BACKWARD, FORWARD AND BACKWARD-FORWARD

DYNAMIC PROGRAMMING MODELS UNDER COMMUTATIVITY CONDITIONS

Sergio Verdú

Department of Electrical Engineering
and Computer Science
Princeton University

H. Vincent Poor

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

1. Introduction

Several authors (Denardo [6], Karp and Held [18], and Bertsekas [3]) have proposed abstract dynamic programming models encompassing a wide variety of sequential optimization problems. The unifying purpose of these models is to impose sufficient conditions on the recursive definition of the objective function to guarantee the validity of the solution of the optimization problem by a dynamic programming iteration. In this paper we propose a general dynamic programming operator model that includes, but is not restricted to, optimization problems. Any functional satisfying a certain commutativity condition (which reduces to the principle of optimality in extremization problems - see Section 2, B2) with the generating operator of the objective recursive 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 distributions in arbitrary probability spaces, iterative computation of stage-separated functions defined on general algebraic systems such as additive commutative semi-groups with distributive products, generation of symbolic transfer functions, and the Chapman-Kolmogorov equations.

Another feature of our operator framework, not shared by previous works, is the ability to formulate forward models completely symmetric to the backward ones. This enables the analysis of open-loop problems (such as those of Propositions 2.1, 2.2 and 2.4) by either approach under the same kind of restrictions (see Sect. 3) and, more significantly, it allows for the formulation of the general backward-forward model.

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 above process, it is more efficient to compute simply the distances of each node from the source and to the destination. As is shown in Section 4, the use of the backward-forward model with nonextremization operators results in interesting applications such as fixed-interval minimum error probability detection in data communications (as discussed in Chapter 2) and the computation of the unconditional transition probabilities of a Markov process.

In Section 2 we present the abstract operator model for finite-horizon backward and forward problems, and several sufficient conditions are shown to ensure the validity of the dynamic programming iteration. In particular, the sufficient condition given by Proposition 2 entails the commutativity of a pair of operators over a certain subset of functions; when applied to infimization problems, this commutativity condition is weaker and not more difficult to check in specific problems than the sufficient conditions previously imposed ([4, Ch. 6]) on the generating operator of the objective recursive function. The use of the proposed operator model is briefly illustrated in Section 2 for optimum stochastic control problems, and in Section 3 for several deterministic problems with extremization and nonextremization operators. The formulation and analysis of the abstract backward-forward operator model are presented in Section 4 along with its application to minimum probability-of-error detection in a general setting.

2. Abstract Finite-Horizon Dynamic Programming Operator Model

Let Q and $T \neq \emptyset$ be arbitrary sets. Consider an operator $V: Q^T \rightarrow Q$, and denote the image of the function $\{q, t \in T\}$ by $Vq_T \in Q$. Let S_i, A_i denote "state" and "action" spaces respectively at stage i , and let M_i be a set of functions ("admissible policies") that map S_i to A_i . Suppose that given $\{L_{\mu_{0,N}, \mu_{0,N}} \in M_{0,N}\}$,¹ a family of functions mapping S_0 to the set Q , and a collection of operators $V_i: Q^{M_i} \rightarrow Q, i=0, \dots, N$, the objective is to find

$$V_{0,N} L_{M_{0,N}}(x) \text{ for every } x \in S_0. \quad (1)$$

In order to solve the above problem via dynamic programming, the following assumptions will be used:

B1. Backward Decomposability of the Objective Function

There exists a collection of operators $H^i: Q^{S_i} \rightarrow Q^{A_i}, i=1, \dots, N$ (mapping functions of states to functions of actions) such that for all $\mu_{0,N} \in M_{0,N}$,

$$L_{\mu_{0,N}} = L_{\mu_{0,N}}^0, \quad (2)$$

$$L_{\mu_{i-1,N}}^{i-1}(x) = H^i L_{\mu_{i,N}}^i(\mu_{i-1}(x)), x \in S_{i-1}; i=1, \dots, N, \quad (3)$$

and

$$L_{\mu_{i,N}}^i: S_i \rightarrow Q; i=0, \dots, N$$

B2. Backward Commutativity of Operators ($VH = HV$)

$$V_{i,N} H^i L_{\mu_{i,N}}^i = H^i V_{i,N} L_{\mu_{i,N}}^i, i=1, \dots, N. \quad (4)^2$$

We then may state the following proposition.

Proposition 1(B)

Define the functions $B^i: S_i \rightarrow Q, i=0, \dots, N$, through the recursion:

$$B^N = V_N L_{M_N}^N, \quad (5.a)$$

and

$$B^{i-1}(x) = V_{i-1} H^i B^i(M_{i-1}(x)) \quad i=1, \dots, N. \quad (5.b)$$

Then, under assumptions B1 and B2, we have

$$V_{0,N} L_{M_{0,N}} = B^0. \quad (6)$$

□

¹The notation $\mu_{i,j} = (\mu_i, \dots, \mu_j), M_{i,j} = M_i \times \dots \times M_j, V_{i,j} = V_i \dots V_j$, is employed.

²The dependence of the functions of actions and states on their arguments is omitted for convenience. However, from the definition of $V_i, i=0, \dots, N$, it is clear that the application of these operators to Q -valued functions is always pointwise.

This work was supported in part by the U.S. Office of Naval Research under Contract N00014-81-K-0014 and by the National Science Foundation under grant ECS-82-12080.

$$V_j H^i L_{\pi M_j}^{i,j} = H^i V_j L_{\pi M_j}^{i,j}, \text{ for all } \pi \in M_{i,j-1}. \quad (8)$$

Proof

Straightforward induction shows that

$$B^i = V_{i,N} L_{M_{i,N}}^i, \quad i=0,\dots,N. \quad (7)$$

□

The fact that the recursions (3) and (5) are defined backwards is only due to the ordering of the operators V in (1). Mere reversal of the stage indices results in a forward solution of $V_{N,0} L_{M_{0,N}}(x)$, $x \in S_N$. The corresponding decomposability and commutativity assumptions are, in this case,

F1. Forward Decomposability of the Objective Function

$$L_{\mu_{0,N}} = L_{\mu_{0,N}}^N \quad (2')$$

where

$$L_{\mu_{0,i+1}}^{i+1}(x) = H^i L_{\mu_{0,i}}^i(\mu_{i+1}(x)), \quad x \in S_{i+1}, i=0,\dots,N-1, \quad (3')$$

and

$$L_{\mu_{0,i}}^i : S_i \rightarrow Q; i=0,\dots,N. \quad H^i : Q^{S_i} \rightarrow Q^{A_{i+1}}, \quad i=0,\dots,N-1,$$

F2. Forward Commutativity of Operators

$$V_{0,i} H^i L_{M_{0,i}}^i = H^i V_{0,i} L_{M_{0,i}}^i, \quad i=0,\dots,N-1. \quad (4')$$

We then have

Proposition 1(F)

Define the functions $F^i : S_i \rightarrow Q$, $i=0,\dots,N$ through the recursion:

$$F^0 = V_0 L_{M_0}^0 \quad (5a')$$

and

$$F^{i+1}(x) = V_{i+1} H^i F^i(M_{i+1}(x)) \quad i=0,\dots,N-1. \quad (5b')$$

Then, under assumptions F1 and F2,

$$V_{N,0} L_{M_{0,N}} = F^N. \quad (6')$$

□

If either pair of assumptions is satisfied for a particular problem, 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 recursions concurrently (Section 4), and because we are interested in recursions which evolve in the direction of the system (for real-time applications, e.g., the Viterbi algorithm).

When this general framework is applied to specific operators H and V , the verification of the commutativity property B2 (or F2) often requires an inductive proof which is common to most problems. Based on such an induction, the next result provides a sufficient condition for B2 (analogously for F2) that entails the verification of the commutativity of H with a single operator V .

Proposition 2

Define the functions $L_{\mu_{i,j}}^{i,j} : S_i \rightarrow Q$, $\mu_{i,j} \in M_{i,j}$, $1 \leq i \leq j \leq N$ through the recursions:

$$L_{\mu_j}^{j,j}(x) = V_{j+1,N} L_{\mu_j M_{j+1,N}}^j(x) \quad (7a)$$

and

$$L_{\mu_{i-1,j}}^{i-1,j}(x) = H^i L_{\mu_{i,j}}^{i,j}(\mu_{i-1}(x)). \quad (7b)$$

Suppose that, for $1 \leq i \leq j \leq N$, we have

Then condition B2 is satisfied.

Proof

Since $L_{\mu_N}^{N,N} = L_{\mu_N}^N$, particularizing (8) for $i=j=N$ results in $V_N H^N L_{M_N}^N = H^N V_N L_{M_N}^N$ (condition B2 for $i=N$). Now suppose that (4) is satisfied for $k+1, \dots, N$. We will show that under condition (8), we have $V_{k,N} H^k L_{M_{k,N}}^k = H^k V_{k,N} L_{M_{k,N}}^k$. The proof will be divided in two stages:

- If $V_{j,N} H^j L_{M_{j,N}}^j = H^j V_{j,N} L_{M_{j,N}}^j$ and condition (8) holds, then $V_j L_{\pi M_j}^{i,j} = L_{\pi}^{i,j-1}$ for $1 \leq i < j$ and for all $\pi \in M_{i,j-1}$.
- If for $k < j \leq N$ and for all $\pi \in M_{k,j-1}$, $V_j L_{\pi M_j}^{k,j} = L_{\pi}^{k,j-1}$ and (8) holds, then $V_{k,N} H^k L_{M_{k,N}}^k = H^k V_{k,N} L_{M_{k,N}}^k$.

Relationship a) can be proved by induction: Let $i=j-1$, then for all $x \in S_i$ and $\pi \in M_i$ we have

$$\begin{aligned} L_{\pi}^{i,j}(x) &= V_{j,N} L_{\pi M_{j,N}}^i(x) = V_{j,N} H^j L_{M_{j,N}}^j(\pi(x)) \\ &= H^j V_{j,N} L_{M_{j,N}}^j(\pi(x)) = H^j V_j L_{M_j}^{i,j}(\pi(x)) = V_j H^j L_{M_j}^{i,j}(\pi(x)) \\ &= V_j L_{\pi M_j}^{i,j}(x), \end{aligned}$$

where the second through sixth equalities follow from (3), (4), (7a), (8), and (7b), respectively. Now suppose that $V_j L_{\pi M_j}^{i+1,j} = L_{\pi}^{i+1,j-1}$ for all $\pi \in M_{i+1,j-1}$ and $i < j-1$, then, for all $x \in S_i$ and $\mu \in M_i$,

$$\begin{aligned} V_j L_{\mu M_j}^{i,j}(x) &= V_j H^{i+1} L_{\mu M_j}^{i+1,j}(\mu(x)) \\ &= H^{i+1} V_j L_{\mu M_j}^{i+1,j}(\mu(x)) \\ &= H^{i+1} L_{\mu}^{i+1,j-1}(\mu(x)) = L_{\mu}^{i,j-1}(x), \end{aligned}$$

where the second equation follows from (8). In order to prove b) it is enough to show that

$$V_N H^k L_{\pi M_N}^k = H^k V_N L_{\pi M_N}^k \quad \text{for all } \pi \in M_{k,N-1}, \quad (9)$$

and

$$V_j H^k V_{j+1,N} L_{\pi M_{j,N}}^k = H^k V_{j,N} L_{\pi M_{j,N}}^k \quad \text{for all } \pi \in M_{k,j-1} \quad (10)$$

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

$$V_{j+1,N} L_{\pi_{k,j} M_{j+1,N}}^k = L_{\pi_{k,j}}^{k,j} \quad \text{for all } \pi_{k,j} \in M_{k,j-1}.$$

So, it suffices to show that

$$V_j H^k L_{\pi M_j}^{k,j} = H^k V_{j,N} L_{\pi M_{j,N}}^k,$$

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

The source of much of the impetus for the above general operator framework is the nice setting due to Bertsekas [3], [4, Part I]. We can underline three main differences between the approach of Bertsekas and that presented here, namely,

- We do not restrict attention to the case $V_i q_T = \inf_{i \in T} q_i$ or $\sup_{i \in T} q_i$. Although, of course, this is the most important case, it is both useful (as illustrated below by several applications) and interesting from an analytical viewpoint to consider nonextremization operators. Note also that since we do not require the operators V_i 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 well-known, cf. [1], [2]).

- ii) Our operator H maps functions of states into functions of actions as in Dynkin-Yushkevich [8]. However, we carry the duality between states and actions one step further by defining the admissible policies as mappings from the state space to the action space rather than to its fibers. The stochastic control formulation of Bertsekas [3] is equivalent to the special case in which the action space is $A_i = S_i \times U_i$, where U_i is a control space and the admissible policies, $\mu \in M_i$ are restricted to satisfy³

$$p_1(\mu(x)) = x, \text{ for all } x \in S_i.$$

Besides its notational convenience, the versatility of the use of general action spaces and action-valued policy functions affords a nice parallelism (e.g., see Section 3.1) between forward and backward problems.

- iii) In [4, Prop. 6.1] several sufficient conditions on the dependence of H on the function of states are shown to guarantee (7) and, hence, the validity of the dynamic programming algorithm in minimization problems satisfying such conditions. Rather, our sufficient condition for the validity of B2, and therefore (7), is the commutativity of the operators V_j and H^i , $i \leq j$, as given by Proposition 2. In general, it is not more difficult to check commutativity directly. Let us focus attention on the special case:

$$Q = \{-\infty\} \cup \mathbb{R} \cup \{+\infty\}, V_i q_T = \inf_{i \in T} q_i, \quad i=0, \dots, N,$$

and consider the following conditions:

- C1) H^i is monotone in the function of states; (in minimization problems, in order to be able to guarantee our commutativity property, it is natural - although not necessary - to impose the monotonicity condition. This is done by Bertsekas [3] and Denardo [6], and appears to have been stated explicitly for the first time by Mitten [17]), and for every $\epsilon > 0$, there exists $\mu^\epsilon \in M_j$ such that

$$H^i L_{\mu^\epsilon}^{i,j} - H^i V_j L_{\mu^\epsilon}^{i,j} \leq \epsilon.$$

- C2) There exists a sequence of policies $\mu^k \in M_j$, $k=0, 1, \dots$, such that

$$L_{\mu^k}^{i,j} \downarrow \inf_{\mu \in M_j} L_{\mu}^{i,j} \text{ and } H^i L_{\mu^k}^{i,j} \downarrow H^i \inf_{\mu \in M_j} L_{\mu}^{i,j}.$$

Conditions C1 and C2 are both implied by the assumptions in [4, Prop. 6.1] and it is elementary to show that (8) follows from either C1 or C2.

In stochastic control problems the key to the existence of μ^ϵ and $\{\mu^k\}$ with the above properties is the fact that the dependence of $L_{\mu}^{i,j}(x)$ on μ is only through $\mu(x)$. Therefore, for every $\delta > 0$ there exists a uniformly δ -optimum policy $\mu^\delta \in A_j^{S_j}$ (not necessarily M_j) such that for all $x \in S_j$

$$L_{\mu^\delta}^{i,j}(x) \leq \begin{cases} V_j L_{\mu^\delta}^{i,j}(x) + \delta & \text{if } V_j L_{\mu^\delta}^{i,j}(x) > -\infty \\ -\frac{1}{\delta} & \text{if } V_j L_{\mu^\delta}^{i,j}(x) = -\infty. \end{cases} \quad (11)$$

In particular cases, such μ^δ can be shown to belong to M_j and mild 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:

$$H^i L(a) = g_{i-1}(a) + \int_{S_i} L(\omega) P_{i-1}(d\omega|a), a \in A_{i-1}$$

$$L^{n+1}(x) = r(x), x \in S_{N+1}.$$

If $A_i \subset S_i \times U_i$ and all admissible policies $\mu \in M_i$ satisfy $p_1(\mu(x)) = x$, for all $x \in S_i$, then it is easy to see that $L_{\mu_{0,N}}^0(x)$ is the expected value of the cost of using $\mu_{0,N} \in M_{0,N}$ when the initial state is $x \in S_0$. To show the existence of $\mu^\delta \in M_j$ satisfying (11), it is enough to assume that the state and control spaces are Borel; A_k , $k=0, \dots, N$ are analytic sets, the costs-per-stage $g_k: A_k \rightarrow \mathbb{R}$, $k=0, \dots, N$ and the terminal cost $r: S_{N+1} \rightarrow \mathbb{R}$ are lower semi-analytic, the transition functions are Borel measurable and $M_j = \{\mu \in A_j^{S_j} \text{ such that } p_1(\mu(x)) = x, x \in S_j \text{ and } \mu \text{ is universally measurable}\}$ (see [5, Cor. 32] and [4, Prop. 50]).

Now, if $V_{j,N} L_{\mu_j,N}^{i,j}(x) > -\infty$ for all $x \in S_j$, then for every $\epsilon > 0$ we can choose $\mu^\epsilon \in M_j$ such that C1 is satisfied. On the other hand, if there exists $\mu \in M_j$ such that $H^i L_{\mu}^{i,j}(a) < +\infty$ for all $a \in A_{i-1}$, then using (11) we can select $\{\mu^k\}$ such that C2 is satisfied. Other stochastic control problems such as those with worst-case, rather than average, objective function and multiplicative and nonnegative, rather than additive costs-per-stage can be shown to satisfy commutativity via conditions C1-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 general framework of Section 2 to classes of backward and forward problems. Because of the causality relation among the state spaces, 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 achievable between the backward and forward formulations, thanks to the versatility of the action spaces and action-valued policies. In Section 3.2 we present applications of nonextremization operators in a very general algebraic setting which includes the conventional finite state-space dynamic programming models. Another example of a nonextremization operator is the Lebesgue integral, which is used in Section 3.3 to derive backward and forward versions of the Chapman-Kolmogorov equations. In this case, the cost-to-go and cost-to-arrive functions are random variables and probability measures, respectively. The corresponding recursions are employed in Section 4 to find the unconditional state transition probabilities of a Markov process using a backward-forward recursion.

3.1. Additive-Cost Deterministic Optimum Control

It is easy to fit the framework of Section 2 to the problem of additive-cost optimum control of a deterministic system $x_{k+1} = f_k(x_k, u_k) \in S_{k+1}$, $u_k \in U_k$, $k=0, \dots, N$; x_0 . It is noteworthy that the dynamic programming formulation presented here allows for a duality between the forward and backward formulations that, unlike previous works (see [11] and [12]), does not impose any restrictions on the system in order to be able to define the forward dynamic programming recursion. For the backward formulation, we make the following identifications:

$$A_i = S_i \times U_i$$

$$H^i L(a) = g_{i-1}(a) + L(f_{i-1}(a))$$

and

$$M_i = \{\mu \in A_i^{S_i} \text{ such that } p_1(\mu(x)) = x \text{ for all } x \in S_i\}.$$

On defining L^i through the recursion (3) - with $L^{N+1} = 0$ -, it is clear that

$$\inf_{\mu_{0,N} \in M_{0,N}} L_{\mu_{0,N}}^0(x) = \inf_{u_{0,N} \in U_{0,N}} \sum_{k=0}^N g_k(x_k, u_k) \text{ s.t. } x_{k+1} = f_k(x_k, u_k), x_0 = x$$

In the forward case, we define

³The projections of a cartesian product are denoted by $p_k: X_1 \times \dots \times X_n \rightarrow X_k$.

$$A_i = S_{i-1} \times U_{i-1}$$

$$H^i L(a) = g_{i+1}(a) + L(p_i(a))$$

and

$$M_i = \{\mu \in A_i^{S_i} \text{ such that } f_{i-1}(\mu(x)) = x \text{ for all } x \in S_i\}.$$

If L^i is defined through (3') - with $L^0 = 0$ -, then it can be checked that

$$\inf_{\mu_{1,N+1} \in M_{1,N+1}} L_{\mu_{1,N+1}}^{N+1}(x) = \inf_{\substack{u_{0,N} \in U_{0,N} \\ \text{s.t.} \\ x_{k+1} = f_k(u_k, u_k); x_{N+1} = x}} \sum_{k=0}^N g_k(x_k, u_k),$$

and in both cases the commutativity property of Proposition 2 is obviously satisfied.

3.2. Recursive Computation of Stage-Separated Functions

Consider an algebraic system $(Q, +, \cdot)$, where $(Q, +)$ is a commutative semi-group and the internal binary operation \cdot is left-distributive with respect to additions. Suppose that $S_i, i=0, \dots, N+1$ are finite sets and that we want to compute

$$B_0(x) = \sum_{x_1 \in S_1} \dots \sum_{x_N \in S_N} g_0(x, x_1) \cdot [g_1(x_1, x_2) \dots [g_N(x_N, x_{N+1})] \dots]$$

for every $x \in S_0$. Some examples of problems of this type are:

- i) Shortest path problems with $(\mathbb{R}, \inf, +)$.
- ii) Computation of Boolean expressions, with $(\{0,1\}, \text{OR}, \text{AND})$.
- iii) Computation of marginal probability distributions, with $(\mathbb{R}^+, +, \cdot)$ (Section 4).
- iv) Symbolic transfer function problems, with $(S, \cup, *)$, where S is the family of sets of finite-length strings of symbols drawn from a finite alphabet, and $*$ denotes string concatenation (see [7] for a generalization of the McNaughton-Yamada algorithm [14] for the computation of regular expressions given arbitrary state graphs, and [18] for optimization problems in finite-state machines).
- v) Dynamic programming for optimization problems where the return space is only partially ordered. If (G, \geq, ρ) is a "regular multiplicative lattice" [16], then (G, \inf) is a commutative semi-group and ρ is distributive with respect to infimization. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition B2.

It is straightforward to use the results of Propositions 1(B) and 2 in order to compute the function B_0 by using dynamic programming. To that end, let $A_i = S_i \times S_{i+1}$, $V_i q_T = \sum q_T$, $M_i = \{\mu \in A_i^{S_i}; \text{ there exists } x^* \in S_{i+1} \text{ such that } \mu(x) = (x, x^*) \text{ for all } x \in S_i\}$ and $H^i L(a) = g_{i+1}(a) \cdot L(p_i(a))$. If the recursion (3) is used to define $L_{\mu_i}^i$, with $L_{\mu_N}^N(x) = g_N(\mu_N(x))$, then it is easy to see that $B_0(x) = V_0 \dots V_N L_{M_0}^0(x)$. The commutativity condition of Proposition 2 follows from the left-distributivity of \cdot with respect to $+$. Analogously, by requiring right-distributivity instead, and interchanging $i-1$ by $i+1$ in the definition of A_i, M_i and H^i above, forward dynamic programming can be used to recursively compute

$$\sum_{x_0 \in S_0} \dots \sum_{x_N \in S_N} [\dots [g_0(x_0, x_1) \cdot g_1(x_1, x_2)] \dots] g_N(x_N, x)$$

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., $q+0 = q, q \cdot 0 = 0 \cdot q = 0$ for all $q \in Q$, then a special case of the above formulation is

$$\sum_{u_0 \in U_0} \dots \sum_{u_N \in U_N} \lambda_0(x, u_0) \cdot [\lambda_1(x_1, u_1) \dots [\lambda_N(x_N, u_N)] \dots]$$

subject to $x_{k+1} = f_k(x_k, u_k), x_0 = x$. To see this, let

$$T_i(x_i, x_{i+1}) = \{u_i \in U_i \text{ such that } f_i(x_i, u_i) = x_{i+1}\}$$

and

$$g_i(x_i, x_{i+1}) = \begin{cases} \sum_{u_i \in T_i(x_i, x_{i+1})} \lambda_i(x_i, u_i) & \text{if } T_i(x_i, x_{i+1}) \neq \emptyset \\ 0 & \text{if } T_i(x_i, x_{i+1}) = \emptyset \end{cases}$$

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

3.3. Discrete-Time Markov Processes: Chapman-Kolmogorov Equations

Consider a Markov process $\{x_t \in \Omega_t, t \in \mathbb{N}\}$ whose finite-dimensional distributions are determined by P_0 , an arbitrary initial probability measure on (Ω_0, F_0) , and by the transition functions $P_i, i=1, \dots$, such that $P_i(x, \cdot)$ is a probability measure on (Ω_i, F_i) , for each $x \in \Omega_{i-1}$ and $P_i(\cdot, B)$ is measurable for each $B \in F_i$. Define

$$P^{i,k}(x, B) = \begin{cases} \int_{\Omega_{i+1,k-1}} \dots \int P_{i+1}(x, d\omega_{i+1}) P_{i+2}(\omega_{i+1}, d\omega_{i+2}) \dots P_k(\omega_{k-1}, B) \\ P_k(x, B) \end{cases}$$

The goal here is to show the Chapman-Kolmogorov equation

$$P^{i,k}(x, B) = \int_{\Omega_j} P^{i,j}(x, d\omega) P^{j,k}(\omega, B), \quad 0 \leq i < j < k, \quad (12)$$

by using the general dynamic programming framework in both backward and forward formulations.

For the backward formulation we make the following identifications: $S_j = \Omega_j; A_j = S_j \times \Omega_{j+1} \times F_{j+1}; M_j = \{\mu \in A_j^{S_j}; \text{ there exists } (\omega, B) \in \Omega_{j+1} \times F_{j+1} \text{ such that } \mu(x) = (x, \omega, B) \text{ for all } x \in S_j\}$, and

$$H^j L(a) = P_j(p_i(a), p_j(a)) L(p_j(a)). \quad (13)$$

If $(\omega, B) \in \Omega \times F$ and $q(\omega, B) = f(\omega) \nu(B)$, then let $V_i q_{\Omega \times F} = \int_{\Omega} f(\omega) \nu(d\omega)$. Fix k and $B \in F_k$, and define $L^j: S_j \rightarrow \mathbb{R}, j < k-1$ through (3), (13) and the initial condition $L^{k-1}(x) = P_k(x, B)$. It can be checked that $P^{i,k}(x, B) = V_{i,k-2} L_{M_{i,k-2}}^{i,k-2}(x), i+1 < k$. The commutativity condition B2 follows trivially from the linearity of the integral. Hence, (7) is satisfied, i.e.,

$$P^{i,k}(x, B) = \int_{\Omega_{i+1}} P_{i+1}(x, d\omega) P^{i+1,k}(\omega, B), \quad i+1 < k,$$

which in turn implies (12). Analogously, for the forward recursion we define $S_j = F_j$ (the cost-to-arrive function is now a probability measure); $A_j = \Omega_{j-1} \times F_{j-1} \times S_j; M_j = \{\mu \in A_j^{S_j}; \text{ there exists } (\omega, B) \in \Omega_{j-1} \times F_{j-1} \text{ such that } \mu(D) = (\omega, B, D), \text{ for all } D \in S_j\}$, and

$$H^j L(a) = P_{j+1}(p_j(a), p_j(a)) L(p_j(a)).$$

Fixing i and $x \in \Omega_i$, and letting $L^{i+1}(B) = P_{i+1}(x, B)$, and proceeding as in the backward case we obtain

$$P^{i,k}(x, B) = \int_{\Omega_{k-1}} P^{i,k-1}(x, d\omega) P_k(\omega, B), \quad i+1 < k$$

which is equivalent to (12). In the more general case, where $\{x_t, t \in \mathbb{N}\}$ is a function of a Markov process, although the Chapman-Kolmogorov equation is not necessarily satisfied [13], it can be shown that the commutativity condition B2 (or F2) is sufficient for its validity.

4. Backward-Forward Finite-Horizon Dynamic Programming

4.1. General Setting and Sufficient Conditions

Given a collection of sets D_i , operators $V_i: Q^{D_i} \rightarrow Q$, $i=1, \dots, N$, and $J_{d_{1,N}} \in Q$ for all $d_{1,N} \in D_{1,N}$ suppose that for all $i \neq j$, $V_i V_j = V_j V_i$ and that $V_{1,N} J_{D_{1,N}}$ can be solved by both backward and forward dynamic programming in the sense that

$$\begin{aligned} V_{1,N} J_{D_{1,N}} &= \bar{V}_{0,N-1} \bar{L}_{\bar{M}_{0,N-1}} \\ &= \bar{V}_{2,N+1} \bar{L}_{\bar{M}_{2,N+1}} \end{aligned}$$

where $\bar{V}_{i+1} = V_i = \bar{V}_{i-1}$; there is a one-to-one mapping from D_i to both policy spaces \bar{M}_{i-1} and \bar{M}_{i+1} ; and $\bar{V}_{0,N-1} \bar{L}_{\bar{M}_{0,N-1}}$ and $\bar{V}_{2,N+1} \bar{L}_{\bar{M}_{2,N+1}}$ satisfy properties B1-B2 and F1-F2 with \bar{S}_0 and \bar{S}_{N+1} equal to singleton sets. Hence, the corresponding cost-to-go and cost-to-arrive functions defined via (5) and (5') satisfy

$$B^i = \bar{V}_{i,N-1} \bar{L}_{\bar{M}_{i,N-1}}^i \text{ and } F^i = \bar{V}_{2,i} \bar{L}_{\bar{M}_{2,i}}^i,$$

respectively.

For each index $i=1, \dots, N$, the above assumptions imply that for every $d_{1,i} \in D_{1,i}$, $J_{d_{1,i}, D_{i+1,N}}$ depends on $D_{i+1,N}$ only through the cost-to-go function B^i , and analogously, for every $d_{i,N} \in D_{i,N}$, $J_{D_{1,i-1}, d_{i,N}}$ depends on $D_{1,i-1}$ only through the cost-to-arrive function F^i . Using both facts and the commutativity of the operators V_j , there exists a function $W_i: Q^{S_i} \times D_i \times Q^{S_i}$ such that

$$V_{1,i-1} V_{i+1,N} J_{D_{1,i-1}, d_{i,N}, D_{i+1,N}} = W_i(F^i, d_i, B^i), \text{ for all } d_i \in D_i. \quad (14)$$

The interest of this result stems from a class of problems where it is desired to compute

$V_{1,i-1} V_{i+1,N} J_{D_{1,i-1}, d_{i,N}, D_{i+1,N}}$ for each $d_i \in D_i$, $i=1, \dots, N$, (e.g., find for each arc in a layered network, the shortest path from source to destination that contains that arc). The straightforward approach would be to repeat a (forward or backward) dynamic programming recursion for each $d_i \in D_i$ and each time index. 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 N . In contrast, if (14) is used, the solution to the above problem requires only two independent (one forward and one backward) dynamic programming recursions. One case of practical interest, where the backward-forward method can be applied, is in the problem of fixed interval minimum error probability detection [19]. It is shown in [19] that backward-forward dynamic programming affords important computational savings over the Hayes-Cover-Riera algorithm [15], which carries out a forward recursion for each decision. The next subsection is devoted to the discussion of the application of backward-forward dynamic programming to optimum data detection in a fairly general framework. It also serves to point out an application of nonextremization operators, namely, the recursive computation of marginal distributions.

4.2. Application to Minimum Probability-of-Error Detection

Consider the following information transmission model which can be shown to include several problems such as multiple-access communications, transmission of convolutionally encoded data and intersymbol interference problems. Let (Ω, \mathcal{F}, P) be a probability space and let $G \subset F$ be the sub- σ -algebra generated by the observation of an F -measurable transformation of a sequence of transmitted symbols $\{u_t \in U_t, t=0, \dots, N-1\}$. As discussed in Chapter 2, optimum decisions based on the *a posteriori* distribution $P^G[u_0, \dots, u_{N-1}]$ can be made according to various optimality criteria; for example, the receiver may select the sequence $u_{0,N-1} \in U_{0,N-1}$ that maximizes $P^G[u_0, \dots, u_{N-1}]$ (globally optimum sequence detection), or the sequence of arguments that maximizes the marginals $P^G[u_i], i=0, \dots, N-1$ (minimum error probability detection). Suppose that there exists a random process $\{X_t: (\Omega, \mathcal{F}) \rightarrow (\Omega_t, \mathcal{F}_t), t=0, \dots, N\}$ such that

M1. X_t is conditionally Markov relative to G .

and

M2. At each stage, there exists a surjective mapping between the process transitions $\Omega_t \times \Omega_{t+1}$ and the alphabet of modulating symbols U_t .

Then, dynamic programming can be used for optimum data demodulation either in forward version (the Viterbi algorithm [9]) for globally optimum sequence detection, or in the backward-forward formulation of this section for minimum error probability detection.

Using assumption M2 above, the minimum error probability detection problem reduces to that of finding the sequence of joint distributions of consecutive states of a Markov process, namely,

$$P^G[x_k \in B_k, x_{k+1} \in B_{k+1}] =$$

$$\int_{\Omega_{0,k-1}} \dots \int_{\Omega_{k-2,k}} P_0^G(d\omega_0) \prod_{l=1}^N P_l^G(\omega_{l-1}, d\omega_l).$$

Let $D_i = \Omega_i \times F_i$ and select

$$d_k = (x_k, C_k) \in D_k \text{ and } d_{k+1} = (x_{k+1}, C_{k+1}) \in D_{k+1}.$$

Then the problem

$$\begin{aligned} J_{D_{0,k-1}, d_{k,k+1}, D_{k+2,N}} &= V_{0,k-1} V_{k+2,N} P_0^G(F_0) P_1^G(\Omega_0, F_1) \dots P_k^G(\Omega_k - \\ &P_{k+1}^G(x_k, C_{k+1}) P_{k+2}^G(x_{k+1}, F_{k+2}) \dots P_N^G(\Omega_N - F_N) \end{aligned}$$

fits the above backward-forward framework with

$$V_i q_{D_i} = \int_{\Omega_i} f(\omega) \nu(d\omega) \text{ for } q_{i, \omega, B} = f(\omega) \nu(B)$$

and $\bar{S}_i = \Omega_i, \bar{A}_i = \Omega_i \times D_{i+1}, \bar{S}_i = F_i, \bar{A}_i = D_{i-1} \times \Omega_i$.

Note the fact that we are fixing two consecutive elements d_k and d_{k+1} , rather than just one, introduces obvious changes in the backward-forward solution. The results of Section 3.3 and Section 4.1 imply that

$$J_{D_{0,k-1}, d_{k,k+1}, D_{k+2,N}} = F^k(C_k) P_{k+1}^G(x_k, C_{k+1}) B^{k+1}(x_{k+1})$$

with

$$F^{i+1}(B) = \int_{\Omega_i} P_{i+1}^G(\omega, B) F^i(d\omega); F^0(B) = P_0^G(B)$$

and

$$B^{i-1}(x) = \int_{\Omega_i} P_i^G(x, d\omega) B^i(\omega); B^{N+1} = 1$$

Therefore, using Fubini's theorem, the sought-after relationship is

$$P^G[x_k \in B_k, x_{k+1} \in B_{k+1}] = \int_{B_k} F^k(dx_k) \int_{B_{k+1}} P_{k+1}^G(x_k, dx_{k+1}) B^{k+1}(x_{k+1}) \quad (15)$$

In data-transmission problems it is common that the alphabets $U_i, i=0, \dots, N$ are finite sets and that

$P^G[u_0, \dots, u_N] = \prod_{k=0}^N \lambda_k(x_k, \mu_k)$ where $x_{k+1} = f_k(x_k, \mu_k)$, x_0 is G -measurable and the following condition is satisfied.

S1. For $k=0, \dots, N$, if there exists $x \in \Omega_k$ and $u, u' \in U_k$ such that $f_k(x, \mu) = f_k(x, \mu')$ then $u = u'$. (Note that condition S1 implies condition M2 above.)

According to the above framework, the only requirement on $(G, +, \cdot)$ for the validity of the backward-forward dynamic programming recursion for the class of problems of Section 3.2, is that $(G, +)$ is a commutative semi-group and that \cdot is distributive with respect to $+$. Furthermore, since in the present case the identity of the addition is an annihilator for multiplication, we can also fit the controlled-state version of the problem (see Section 3.2), and the following scheme for the computation of the marginals can be used.

$$\begin{aligned}
 P^G[u_i] &= \sum_{u_{0,t-1} \in U_{0,t-1}} \sum_{u_{t+1,N} \in U_{t+1,N}} \prod_{k=0}^N \lambda_k(x_k, \mu_k) \\
 &= \sum_{\substack{x_t, x_{t+1} \in \Omega_t \times \Omega_{t+1} \\ \text{s.t.} \\ x_{t-1} = f_t(x_t, u_t)}} F^t(x_t) \lambda_t(x_t, \mu_t) B^{t+1}(x_{t+1}) \tag{16}
 \end{aligned}$$

with

$$F^{k+1}(x) = \sum_{\substack{x_k \in \Omega_k \\ \text{s.t. there exists} \\ u \in U_k, f_k(x_k, u) = x}} F^k(x_k) \lambda_k(x_k, \mu_k),$$

$$B^k(x) = \sum_{u_k \in U_k} B^{k+1}(f_k(x, \mu_k)) \lambda_k(x, \mu_k);$$

and

$$F^0(x) = \begin{cases} 1 & x = x_0 \\ 0 & x \neq x_0 \end{cases}, \quad B^{M+1} = 1$$

Moreover, if the following condition is satisfied

S2. For $k=0, \dots, N$, if there exists $x, x' \in \Omega_k$ and $u, u' \in U_k$ such that $f_k(x, \mu) = f_k(x', \mu')$ then $u = u'$.

(e.g., frequently $f_k(\cdot, \cdot)$ $k=0, \dots, N$ is a shift register system); then (16) is further reduced to

$$\begin{aligned}
 P^G[u_i] &= \sum_{\substack{x_{t+1} \in \Omega_{t+1} \\ \text{s.t. there exists} \\ \lambda \in \Omega_t, x_{t+1} = f_t(\lambda, u_t)}} F^{t+1}(x_{t+1}) B^{t+1}(x_{t+1}). \tag{17}
 \end{aligned}$$

- [1] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, New York: Academic Press, 1982.
- [2] R. Bellman, *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.
- [3] D. P. Bertsekas, "Monotone Mappings with Application in Dynamic Programming," *SIAM J. Control*, vol. 15, no. 3, pp. 438-464, May 1977.
- [4] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The Discrete Time Case*, New York: Academic Press, 1978.
- [5] D. Blackwell, D. Freedman and M. Orkin, "The Optimal Reward Operator in Dynamic Programming," *Ann. Probability*, vol. 2, no. 5, pp. 926-941, October 1974.
- [6] E. V. Denardo, "Contraction Mappings in the Theory Underlying Dynamic Programming," *SIAM Rev.*, vol. 9, no. 2, pp. 165-177, April 1967.
- [7] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, MA: Addison-Wesley P.C., 1974.
- [8] E. B. Dynkin and A. A. Yushkevich, *Controlled Markov Processes*, New York: Springer-Verlag, 1979.
- [9] G. D. Forney, "The Viterbi Algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268-278, March 1973.
- [10] A. Kauffmann and R. Cruon, *Dynamic Programming: Sequential Scientific Management*, New York: Academic Press, 1977.
- [11] R. E. Larson, *State Increment Dynamic Programming*, New York: Elsevier, 1968.
- [12] R. E. Larson and J. L. Casti, *Principles of Dynamic Programming, Part I*, New York: Marcel Dekker, 1978.
- [13] M. Rosenblatt, *Markov Processes: Structure and Asymptotic Behavior*, New York: Springer-Verlag, 1971.
- [14] R. McNaughton and H. Yamada, "Regular Expressions and State Graphs for Automata," *IRE Trans. Computers*, vol. 9, no. 1, pp. 39-47, January 1960.
- [15] J. F. Hayes, T. M. Cover and J. B. Riera, "Optimal Sequence Detection and Optimal Symbol-by-Symbol Detection: Similar Algorithms," *IEEE Trans. Comm.*, vol. COM-30, no. 1, pp. 152-157, January 1982.
- [16] T. A. Brown and R. E. Strauch, "Dynamic Programming in Multiplicative Lattices," *J. Math. Analysis and Applications*, vol. 12, pp. 364-370, 1965.
- [17] L. G. Mitten, "Composition Principles for Synthesis of Optimum Multi-Stage Processes," *Operations Research*, vol. 12, no. 4, pp. 610-619, 1964.
- [18] R. M. Karp and M. Held, "Finite-State Processes and Dynamic Programming," *SIAM J. Appl. Math.*, vol. 15, no. 3, pp. 693-718, May 1967.
- [19] S. Verdú "On Fixed-Interval Minimum Symbol Error Probability Detection," Coordinated Science Laboratory, Urbana, IL, TR-990, UILU-ENG83-2211, June 1983.