TP5 - 3:00

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 [Derman 6], [Karp and Held 14], and Bertman [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 limited to, optimization problems. Any functional satisfying a certain commutativity condition (which reduces to the principle of optimality in extremization problems—see Section 2.1, B2) with the generating operator of the objective recursive function results in a sequential problem solvable by a dynamic programming iteration. Examples of sequential nonoptimization problems fitting this framework are the derivation of marginal distributions in arbitrary probability spaces, iterative computation of stage-sequential functions defined on general algebraic systems such as additive commutative semi-groups with distributive products, generation of symmetrical transfer functions, and the Chapman-Kolmogorov equation.

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 importantly, it allows for the formulation of the general backward-forward model.

The backward-forward model presented in this paper is derived 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 this end, consider the following simple example of this method. In a layered network, the shortest path consisting of a particular arc can obviously be obtained by deleting any other arc in the same layer and solving for the shortest route by either forward or backward dynamic programming; however, if the path must be solved for on an arc in the network, then, rather than repeating the above process, it is more efficient to compute simply the taxation of each node from the source and to the destination. As is shown in Section 4, the use of the backward-forward model with nonmonotone operator results in interesting applications such as fixed-accuracy minimum error probability detection in data communication (as discussed in Chapter 2) and the computation of the unconditional transition probability of a Markov process.

In Section 2, we present the abstract operator model for horizon backward and forward problems, and several sufficient conditions are shown to ensure the validity of the dynamic programming iteration. In particular, the sufficient conditions given by Proposition 2 entail the commutativity of a pair of operators over a certain subset of functions: when applied to informative problems, this commutativity condition is weaker and not more difficult to check in specific problems than the sufficient conditions previously imposed [4, Ch. 4] on the generating operator of the objective recursive function. The formulation and analysis of the abstract backward-forward operator model presented in Section 4 along with its applications to minimum probability-of-error decision in a general setting.

This work was supported in part by the U.S. Office of Naval Research under Contracts N00014-81-K-0013 and the National Science Foundation under Grant ECS-82-12180.

2. Abstract Finite-Horizon Dynamic Programming Operator Model

Let \( Q \) and \( P^{(a)} \) be arbitrary sets. Consider an operator \( V: Q \times \mathcal{A} \rightarrow \mathbb{R} \), and denote the image of the function \( f(x,a) \) by \( V_x a \). Let \( \mathcal{A} \) denote "state" and "action" space respectively at stage 1, and let \( M_0 \) be a set of functions ("observables" pertinent) that map \( S \), to \( \mathcal{A} \). Suppose that given \( (f_1, f_2, \ldots, f_N) \), a family \( \{ f(x,a) \} \) of functions mapping \( S \) to \( Q \), and a collection of operators \( V: Q \times \mathcal{A} \rightarrow \mathbb{R} \), then the objective is to find

\[
\min_{a} V_0 I_{M_0} (x) \quad \text{for every } x \in S.
\]

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 \( B_i: Q \rightarrow Q, i=1, \ldots, N \) (mapping states to states of functions of actions) such that for all \( a \in M_0 \)

\[
\begin{align*}
L_{\mathcal{A}_0} (f) &= \min_{a} \min_{x} V(x,a) \quad \text{(2)} \\
L_{\mathcal{A}_1} (f) &= \min_{a} \min_{x} B_i(f(x,a)) \quad \text{x } S_i, \quad i=1, \ldots, N, \quad \text{(3)} \\
L_{\mathcal{A}_N} : S_i \rightarrow Q \quad \text{i=0, N}.
\end{align*}
\]

B2. Backward Commutativity of Operators (\( B_i = B_i^T \))

\[
\begin{align*}
V_i S_i B_i L_{\mathcal{A}_i} &= V_i S_i B_i L_{\mathcal{A}_i}, \\
\text{and} \\
B_i (x) &= V_i S_i H_i B_i (M_i (x)), \quad i=1, \ldots, N, \quad \text{(5)}
\end{align*}
\]

We then may state the following proposition.

Proposition 1(B)

Define the functions \( B_i: Q \rightarrow Q, i=1, \ldots, N \), through the recursion:

\[
\begin{align*}
B_0 &= V_0 S_0 L_{\mathcal{A}_0}, \\
B_i (x) &= V_i S_i H_i B_i (M_i (x)), \quad i=1, \ldots, N, \quad \text{(6)}
\end{align*}
\]

Then, under assumptions B1 and B2, we have:

\[
V_0 S_0 L_{\mathcal{A}_0} = B_0.
\]

\[\text{This work was supported in part by the U.S. Office of Naval Research under Contract N00014-81-K-0013, and by the National Science Foundation under Grant ECS-82-12180.}\]

\[\text{The author is grateful to Professors A. A. Goldsmith and C. B. Papadimitriou for discussions concerning the problem.}\]
Proof

Straightforward induction shows that
\[ V_i^{H} = V_i^{H} V_{i+1}^{H}, \quad i = 0, \ldots, N. \] (7)

The fact that the recurrences (3) and (5) are defined backwards is only due to the ordering of the operators \( V_i \) in (1). More reversal of the stage indices results in a forward solution of \( V_i^{H} V_{i+1}^{H} = V_{i+1}^{H} V_i^{H} \) for \( i = 0, \ldots, N-1 \). The corresponding decomposability and commutativity assumptions are true in this case.

**F1. Forward Decomposability of the Objective Function**

Define the functions \( F : S \times Q \times i = 0, \ldots, N \) through the recursion:
\[ F^{i+1}(x) = V_i^{H} F^{i}(V_i(x)), \quad i = 0, \ldots, N-1. \] (5a)

and
\[ F^{0}(x) = V_0^{H} \phi(x). \] (5b)

Then, under assumptions F1 and F2,

\[ V_i^{H} V_{i+1}^{H} = F^{i+1}(V_i(x)) \] (8)

We then have

**Proposition F2**

Define the functions \( F : S \times Q \times i = 0, \ldots, N \) through the recursion:
\[ F^{i+1}(x) = V_i^{H} F^{i}(V_i(x)), \quad i = 0, \ldots, N-1. \] (5a)

and
\[ F^{0}(x) = V_0^{H} \phi(x). \] (5b)

Then, under assumptions F1 and F2,

\[ V_i^{H} V_{i+1}^{H} = F^{i+1}(V_i(x)) \] (8)

If either part of the 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 stating that the other one is backwards, we choose to maintain always the original indices and present both the Forward and the Backward formulations. This is due to 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 for F2 (or F2) then requires an inductive proof which is common to most problems. Based on such an induction, the next result provides a sufficient condition for B2 (anaolgously for F2) that entails the verification of the commutativity of \( H \) with a single operator \( V \).

**Proposition 2**

Define the functions \( L_{i,j}^{V}(x) : S \times Q \times i = 0, \ldots, M_j \) through the recursion:
\[ L_{i+1,j}^{V}(x) = V_{i,j}^{H} L_{i+1,j}^{V}(x) \] (9a)

and
\[ L_{i,j}^{V}(x) = H^{i} L_{i,j}^{V}(x) \] (9b)

Suppose that, for \( 1 \leq j \leq N \), we have

\[ V_i^{H} L_{i,j}^{V} = H^{i} V_i^{H} L_{i,j}^{V}, \quad i = 0, \ldots, N. \] (10)

Then condition B2 is satisfied.

**Proof**

Since \( L_{i,j}^{V} = L_{i,j}^{V} \), particularizing (8) for \( i = 0, \ldots, N \) results in \( V_i^{H} V_{i+1}^{H} L_{i,j}^{V} = H^{i} V_i^{H} V_{i+1}^{H} L_{i,j}^{V} \) (condition B2 for \( i = 0, \ldots, N \)) and, now suppose that (4d) is satisfied for \( i = 0, \ldots, N \). We will show that under condition (8), we have \( V_i^{H} L_{i,j}^{V} = H^{i} V_i^{H} L_{i,j}^{V} \). The proof will be divided into two stages:

i) If \( V_i^{H} L_{i,j}^{V} = H^{i} V_i^{H} L_{i,j}^{V} \), and condition (8) holds, then \( V_i^{H} L_{i+1,j}^{V} = H^{i} V_i^{H} L_{i+1,j}^{V} \) for \( i = 0, \ldots, N \) and for all \( 1 \leq j \leq N \).

ii) If \( V_i^{H} L_{i,j}^{V} = H^{i} V_i^{H} L_{i,j}^{V} \) and for all \( 1 \leq j \leq N \), then \( V_i^{H} L_{i+1,j}^{V} = H^{i} V_i^{H} L_{i+1,j}^{V} \).

Relationship (i) can be proved by induction: Let \( i = 0, \) then for all \( 1 \leq j \leq N \) and \( 1 \leq k \leq M_j \), we have

\[ L_{i,j}^{V}(x) = V_i^{H} L_{i,j}^{V}(x) = V_i^{H} H^{i} V_i^{H} L_{i,j}^{V}(x) = V_i^{H} H^{i} L_{i,j}^{V}(x) \] (11)

where the second equality is justified in (3), (4d), (7a), (8), and (10), respectively. Now suppose that \( V_i^{H} L_{i,j}^{V} = H^{i} V_i^{H} L_{i,j}^{V} \) for all \( 1 \leq j \leq N \) and \( i = 0 \), then, for all \( 1 \leq j \leq N \) and \( \mu \leq M_j \),

\[ V_i^{H} L_{i+1,j}^{V}(x) = V_i^{H} H^{i} L_{i+1,j}^{V}(x) \] (12)

and

\[ H^{i} V_i^{H} V_{i,j}^{H} L_{i+1,j}^{V}(x) \] (13)

so that it suffices to show that

\[ V_i^{H} L_{i+1,j}^{V}(x) = H^{i} V_i^{H} L_{i+1,j}^{V}(x) \] (14)

to satisfy the induction proof (8). In order to prove b) it is enough to verify that

\[ V_i^{H} V_{i,j}^{H} L_{i+1,j}^{V}(x) = H^{i} V_i^{H} V_{i,j}^{H} L_{i+1,j}^{V}(x) \] (15)

and

\[ H^{i} V_i^{H} V_{i,j}^{H} L_{i+1,j}^{V}(x) \] (16)

to satisfy the induction proof (8). In order to prove b) it is enough to verify that

\[ V_i^{H} L_{i+1,j}^{V}(x) = H^{i} V_i^{H} L_{i+1,j}^{V}(x) \] (17)

and

\[ H^{i} V_i^{H} L_{i+1,j}^{V}(x) \] (18)

so that it suffices to show that

\[ V_i^{H} L_{i+1,j}^{V}(x) = H^{i} V_i^{H} L_{i+1,j}^{V}(x) \] (19)

but this readily follows from (10) and (7).

The proof of the main result is completed by verifying the commutativity property B2 for the above general operator framework. The proof is as follows (see [1], [3].) We can simplify the main result of the above general operator framework by the approach of Bertsekas [1, Part 4]. We can eliminate the main difference between the approach of Bertsekas and that presented here, namely,

\[ V_i^{H} L_{i,j}^{V}(x) = H^{i} V_i^{H} L_{i,j}^{V}(x) \] (20)

and

\[ H^{i} V_i^{H} L_{i,j}^{V}(x) \] (21)

in the respective sections.

In particular, we prove the following

\[ V_i^{H} L_{i,j}^{V}(x) = H^{i} V_i^{H} L_{i,j}^{V}(x) \] (22)

for all \( 1 \leq j \leq N \) and \( 1 \leq i \leq N-1 \).
results in low entropy if the condition is met.

Subsequent to this, we denote the expected value of the relevant policy functions under the steady-state distribution $\pi$ by $\mathbb{E}_\pi$.

We then consider

$$p^*(\alpha) = x, \quad \text{for all } x \in \mathcal{X},$$

and

$$p^*(\alpha) = x, \quad \text{for all } x \leq x_0,$$

where $x_0$ is chosen to be the solution of

$$\mathbb{E}_\pi \left[ \sum_{\alpha \in \mathcal{A}} r(x, \alpha) p^*(\alpha) \right] = 0.$$

It then follows that

$$p^*(\alpha) = x, \quad \text{for all } x \leq x_0,$$

so that $\mathbb{E}_\pi$ is a constant function for $x < x_0$. The dynamics of the system can then be described by

$$\mathbb{E}_\pi \left[ \sum_{\alpha \in \mathcal{A}} r(x, \alpha) p^*(\alpha) \right] = 0.$$

By applying the principle of optimality, we obtain

$$p^*(\alpha) = x, \quad \text{for all } x \leq x_0,$$

where $x_0$ is chosen to be the solution of

$$\mathbb{E}_\pi \left[ \sum_{\alpha \in \mathcal{A}} r(x, \alpha) p^*(\alpha) \right] = 0.$$

If $A \leq C \leq S \leq \mathcal{U}$, then it follows that

$$p^*(\alpha) = x, \quad \text{for all } x \leq x_0,$$

so that $\mathbb{E}_\pi$ is a constant function for $x < x_0$. The dynamics of the system can then be described by

$$\mathbb{E}_\pi \left[ \sum_{\alpha \in \mathcal{A}} r(x, \alpha) p^*(\alpha) \right] = 0.$$

By applying the principle of optimality, we obtain

$$p^*(\alpha) = x, \quad \text{for all } x \leq x_0,$$

where $x_0$ is chosen to be the solution of

$$\mathbb{E}_\pi \left[ \sum_{\alpha \in \mathcal{A}} r(x, \alpha) p^*(\alpha) \right] = 0.$$

If $A \leq C \leq S \leq \mathcal{U}$, then it follows that

$$p^*(\alpha) = x, \quad \text{for all } x \leq x_0,$$

so that $\mathbb{E}_\pi$ is a constant function for $x < x_0$. The dynamics of the system can then be described by

$$\mathbb{E}_\pi \left[ \sum_{\alpha \in \mathcal{A}} r(x, \alpha) p^*(\alpha) \right] = 0.$$
subject to $x_{i1} = f_i(x_{i2}, a_i), x_{i2} = x$. To see this, let

$$T_i(x_{i1}, a_i, u_i) = \sum_j x_{i2, j}(a_i)$$

and

$$M = \{a \in A^I : \text{s.t. } f_i(x_{i2}, a_i) = x \text{ for all } x \in S_i \}.$$  

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

$$M = \{a \in A^I : \text{s.t. } f_i(x_{i2}, a_i) = x \text{ for all } x \in S_i \}.$$ 

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

3.2. Recursive Computation of Stage-Separable Functions

Consider an algebraic system $(Q, \cdot, +, \cdot, \cdot)$, where $(Q, \cdot, +)$ is a commutative semi-group and the internal binary operation $\cdot$ is left-distributive with respect to addition. Suppose that $S_i, A_i, D_i, N_i$ are finite sets and that we want to compute

$$B_i(x) = \sum_{x_{i1} \in A_i} \sum_{x_{i2} \in D_i} g_i(x_{i1}, x_{i2}) f_i(x_{i2}, a_i).$$

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

i) Shortest path problems with $B_i(x)$, inf.
ii) Computation of Boolean expressions, with $B_i(x)$, OR, AND.
iii) Computation of marginal probability distributions, with $B_i(x)$, $\cdot$, $\cdot$ (Section 4.1).

3.3. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.4. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.5. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.6. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.7. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.8. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.9. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.10. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.11. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.12. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

3.13. Dynamic Programming for Optimization Problems where the return space is only partially ordered. If $(Q, \cdot, +)$ is a regular multiplicative lattice [16], then $(Q, \cdot, +)$ is a commutative semi-group and is distributive with respect to infimum. Interestingly, the version of the optimality principle of [16] is a special case of the commutativity condition $B_i(x), a_i$.

4.3. General Setting and Subsequent Conditions

Given a collection of sets $D_i$, operators $V_i$, $Q_i$, $O_i = N_i - 1$, and $J_{i-1}X_i$ for all $i \geq 0$ in $D_0$, suppose that for all $i \geq 1, V_iV_{i+1} = V_i$ and that $J_{i-1}X_i$ can be solved by both backward and forward dynamic programming in the same that

$$V_i = J_{i-1}X_i = v_i = V_{i+1}J_{i-1}X_i = V_{i+1}v_{i+1},$$

where $v_i = V_iV_{i+1} = 1$ is a one-to-one mapping from $D_i$ to both policy spaces $R_i$ and $R_{i-1}$ and $V_iV_{i+1}$ and $V_{i+1}v_{i+1}$ equal to starred sets. Hence, the corresponding cost-to-go and cost-to-arrive functions defined via (54) and (57) satisfy

$$B_i = V_iL_iU_iB_i \quad \text{and} \quad F_i = V_iL_iU_iF_i,$$

respectively.

For each index $i \in \{1, N\}$, the above assumptions imply that for every $i \in D_i, J_{i-1}X_i$, depends on $D_{i-1}$ only through the cost functions $B_i, F_i$, and analogously, for every $i \in D_i, J_{i-1}X_i$, depends on $D_i$ only through the cost-to-arrive function $F_i$. Using both facts and the commutativity of the operators $V_i$, there exists a function $W_i : Q_i \times L_i \times D_i \rightarrow Q_i \times L_i$ such that

$$V_i = V_{i+1}j_i = v_{i+1} = W_i(B_i, F_i, D_i)$$

for all $i \in D_i, 1 \leq i \leq N$. The interest of this result stems from a class of problems where it is desirable to compute

$$V_i = V_{i+1}j_i = v_{i+1} = W_i(B_i, F_i, D_i),$$

for each $i \in D_i, 1 \leq i \leq N$, (14) and (15) for all $i \in D_i, 1 \leq i \leq N$. The interest of this result stems from a class of problems where it is desirable to compute

$$V_i = V_{i+1}j_i = v_{i+1} = W_i(B_i, F_i, D_i),$$

for each $i \in D_i, 1 \leq i \leq N$, (14) and (15) for all $i \in D_i, 1 \leq i \leq N$. The interest of this result stems from a class of problems where it is desirable to compute

$$V_i = V_{i+1}j_i = v_{i+1} = W_i(B_i, F_i, D_i).$$

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. It is shown in [19] that backward-forward dynamic programming affords important computational savings over the Hayman-Cover-Rata algorithm [15], which comes out a forward recursion for each decision. The subsection is devoted to the discussion of the application of backward-forward dynamic programming to optimum data detection in a facility general framework. It also serves to point out an application of nonsequential operators, namely, the recursive commutativity of marginals 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 convolutedly encoded data and interdependent interference problems. Let $(D,F,P)$ be a probability space and let $G \in C_1$ be the sub-algebra generated by the observation of an $F$- measurable transformation of a sequence of transmitted symbols $(a_1,a_2,\ldots,a_{N-1})$. As discussed in Chapter 2, optimum decisions based on the a posteriori distribution $P^n[a_1,a_2,\ldots,a_{N-1}]$ can be made according to various optimality criteria for example, the receiver may select the sequence $a_1,a_2,\ldots,a_{N-1}$ that maximizes $P^n[a_1,a_2,\ldots,a_{N-1}]$ (global optimum sequence decision), or the sequence of arguments that maximizes the marginal $P^n[a_1] = N_{(N-1)}$ (minimum error probability decision). Suppose that there exists a random process $(X_0,X_1,\ldots,X_N)$ such that $X_0, X_1, \ldots, X_N$ is conditionally Markov relative to $G$. and

$$P^n[a_1,a_2,\ldots,a_{N-1}] \quad \text{for all} \quad a_1,a_2,\ldots,a_{N-1} \in N.$$
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 \(-a\) is distributive with respect to \(\cdot\). Furthermore, since in the present case the identity of the addition as an

\[
P^*(a) = \sum_{i=1}^{n} \sum_{j=1}^{m} \prod_{k=1}^{m} k_j(a_j, a_k) = \sum_{i=1}^{n} \sum_{j=1}^{m} \prod_{k=1}^{m} F(x_i, b_i, x_k, a_k)
\]

(16)

with

\[
F^*(a) = \sum_{i=1}^{n} \sum_{j=1}^{m} \prod_{k=1}^{m} F(x_i, b_i, x_k, a_k)
\]

\[
B^*(a) = \sum_{i=1}^{n} \sum_{j=1}^{m} B^*(a) \prod_{k=1}^{m} F(x_i, b_i, x_k, a_k)
\]

and

\[
P^*(a) = \begin{cases} 1 & x = x_0 \in U \setminus A \backslash A \setminus B, B^*(a) = 1 \\ 0 & x \neq x_0 \end{cases}
\]

Moreover, if the following condition is satisfied

52. For \(a \in \mathcal{D}_N\), if there exists \(x, y \in U_1\) and \(u, u' \in U_2\), such that

\[
f(x, u) = f(y, u')
\]

it is further reduced to

\[
P^*(a) = \sum_{x \in \mathcal{D}_N} P^*(x) \prod_{i=1}^{m} F(x_i, b_i, x_k, a_k)
\]

(17)

REFERENCES