JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 29, 523–557 (1970)

The Concept of "State" in Discrete Dynamic Programming*

SALAH E. ELMAGHRABY

North Carolina State University, Raleigh, North Carolina 27607

Submitted by Richard Bellman

Dynamic programming problems have always been stated in terms of stages, states, decisions, rewards, and transformations. This paper investigates the possibility of starting the analysis with a much simpler, i.e., coarser, structure than is commonly required. It establishes necessary and sufficient conditions for the analysis of such generally stated problems by the methodology of dynamic programming.

INTRODUCTION

From its very beginnings dynamic programming (DP) problems have always been cast, in fact, *defined*, in terms of:

(i) A physical process which progresses in *stages*.

(ii) At each stage, the physical system is characterized by a (hopefully small) set of parameters called the *state* variables.

(iii) At each stage, and for any given state, we have a choice of a number of *decisions* (which may depend on both stage and state); assuming that the past history of the system is of no importance in determining current or future decisions,

(iv) The effect of a decision is twofold: (a) A "*reward*" and (b) a *transformation* of the state variables. Both reward and transformation are known functions (i.e., single-valued mappings) of the input state and the decision into the real line and the state space, respectively.[1]

(v) The purpose of the process is to maximize (or minimize) some function of the state and decision variables.

* This research was partially supported by NSF grant GK 2647.

[1] If the reward is a function of the input state, the decision made, as well as the output state, then the transformation function (that defines the output state as a function of the input state and the decision made) is used to modify the reward function to be a function of the input state and decision only.

(See Bellman's book [2] p. 81; Nemhauser's book [10] p. 22; Mitten [9]; Mitten and Dernado [4], and all the literature on Markov DP, e.g., the paper by Brown [3] and the references cited therein.)

The first hint that such a blanket assumption concerning the knowledge of the state space and the transformation function (i.e., the mapping of the state space into itself under an input decision) may not be a valid assumption was given by Karp and Held [7] in their 1967 paper. Unfortunately, the hint was almost lost in the maze of terminology applicable to the theory of automata, and in the focusing of their attention on *decisions*, *policies*, and *policy spaces* instead of *states*, *transformations*, and *state spaces*. The central role that the concept of state plays in the analysis was, at best, partially obscured if not completely lost.

The purpose of the analysis presented below is to answer the following question: the overwhelming majority of decision processes (for optimization purposes) exist, i.e., are *defined*, with much less structure than that demanded by the model requiring the structure given in (i) to (v) above. Can one analyze such processes? In other words, can one start with a more abstract (i.e., less structured) model and *deduce from it* the structure needed for a DP formulation? Put in a still different way, the question may be phrased as follows: if one starts with less information than that demanded by the structure of (i)–(v) above, what are the necessary and sufficient conditions for deducing such a structure from the given data?

The answer to the first statement of the question is in the affirmative. We shall draw upon certain notions developed in Control Theory, and in particular on the excellent and highly organized work of Zadeh, reported in the book by Zadeh and Desoer [14], Chapters 1–3, and as Chapter 3 in the book edited by Mesarovic [8].

It turns out that the information available on operational systems define what we shall refer to (following Karp and Held) as a *decision process*. Since we focus our attention on discrete processes, we shall be concerned only with *discrete decision processes* (DDP) in this paper. (Although we use the same terminology as Karp and Held because of its descriptive appeal, our DDP is quite different from theirs.) By "parameterizing" the DDP in a particular fashion, we transform a DDP into a *sequential discrete decision process* (SDDP) which possesses the detailed structure of (i)–(v) above. As is well known, this does not yet "make" a dynamic programming model; to it we must add Mitten's monotonicity condition [9] to complete the picture.

All the above notions will be made precise below. But first we illustrate by several examples our contention that "the overwhelming majority of decision processes (for optimization purposes) occur *without* the structure of state space and state transition function already defined." These are well-known examples in the literature of Operations Research and some are of

classical interest. Reference will be made to these examples throughout the remainder of the paper.

EXAMPLES

1. *The Traveling Salesman Problem*

A salesman is stationed in some city which we shall refer to as city $\underline{1}$, and must visit each of $n-1$ other cities which are referred to as cities $\underline{2}$, $\underline{3}$,..., $\underline{n}$ once (and only once) and return at the end of the tour to city $\underline{1}$. The time he spends in each city is fixed and, therefore, does not influence the optimizing policy. The distance between any pair of cities $\underline{i}$ and $\underline{j}$ is a known constant, d_{ij} ; and it is possible that $d_{ij} \neq d_{ji}$; i.e., the square matrix of distances need not be symmetric.

Determine the sequence (of cities visited) which minimizes his total distance traveled in a complete tour.

Notice that a *feasible* schedule (i.e., a feasible policy) is any permutation of the n numbers which starts with $\underline{1}$, ends with $\underline{1}$, and each $\underline{i}$ in the set $B = \{2, 3,..., n\}$ appears once and only once. There are exactly $(n-1)!$ such feasible sequences. Given any sequence (i.e., policy) π, its "value" is immediately determined from

$$v(\pi) = \sum_{k=1}^{n-1} d_{i_k, i_{k+1}} + d_{1,i_1} + d_{i_n,1}, \qquad i_k \in B. \tag{1}$$

2. *The "Faithful" Traveling Salesman Problem*

This is an interesting variation on the Traveling Salesman Problem stated in 1. The statement of the problem is the same except that, being a faithful husband and a good father, the salesman insists on spending weekends at home. Consequently, he cannot be away from city $\underline{1}$ for more than 5 days, and it is assumed that once he returns to city $\underline{1}$ (even before the weekend), he does not leave before the following Sunday night. For the sake of simplicity, we assume further that he spends one day in any city $\underline{i} \in B$. His weekly base salary is C per week and cost of travel is proportional to the distance traveled.

What is the schedule that minimizes his total travel and salary cost?

Clearly, a feasible schedule is any permutation of the numbers in the set $BU\{1\}$ such that it starts with $\underline{1}$, ends with $\underline{1}$, each $\underline{i} \in B$ appears once and only once, and no more than 5 cities appear between any two 1's. Needless to say, the number of feasible schedules is much larger than in the case of the

Traveling Salesman Problem. Still, given any feasible schedule, its value can be determined from

$$v(\pi) = v(1i_1i_2 \cdots i_r1) = \sum_r c_{i_r,i_{r+1}} + CM, \tag{2}$$

where $M + 1$ is the number of 1's in the schedule $1i_1i_2 \cdots i_r1$, and $c_{i_r,i_{r+1}} = d_{i_r,i_{r+1}}$.

3. *A Dynamic Economic Manufacturing Quantity Problem*

A plant produces one product (e.g., an ice cream factory) whose demand r_t is known for a finite planning horizon $t = 1, 2,..., T$. The initial inventory of the product I_0, as well as the desired terminal inventory, I_T, are also known. The management of the stocks of the product involves two costs: a setup cost c_0 each time the facility is geared for production, and an inventory cost h per unit of end-of-period inventory.

What is the production schedule which minimizes the total cost of setup and inventory charges and satisfies the demand with no delay?

It is intuitively obvious, and, in any case, can be easily shown, that one must first "net out" the initial inventory against the demand. Then, if x_t is the quantity produced in period t, we have that $x_tI_{t-1} = 0$ for all $t = 1, 2,..., T$. It follows immediately that there are exactly 2^{T-1} different feasible schedules of production among which the optimal schedule must lie (these are precisely the schedules which produce whole requirements). Clearly, given any schedule, one can easily evaluate its value by

$$c_0n + h \sum_{t=1}^{T-1} I_t$$

where n is the number of setups called for by the schedule.

4. *A Module Placement Problem*

We are given n positions, indexed 1, 2,..., n, equally spaced along a line, and n objects similarly indexed, one of which is to be placed in each position. Let a_{ij} denote the number of connections between object i and object j.

Assign objects to positions so as to minimize the sum of the lengths of all connections, where the length of a connection between positions i and j is $|i - j|$.

There are $n!$ feasible assignments of the n objects to the n positions. Given any assignment, its value can be evaluated from

$$\sum_{\{i,j|i<j\}} a_{p(i),p(j)}(j - i),$$

where $p(i)$ is the object in position i and $p(j)$ is the object in position j.

In each of the above examples, the problem is stated in terms of an input string of decisions, the *input policy*, or simply the *policy*, and the *output* (or *response*) *value*, or simply the *value*. In its natural context, the concept of "state" is foreign to the operational system. We superimpose such a concept on the basic structure of the process in order to facilitate analysis and reduce the computing burden to manageable proportions.

The remainder of this paper is concerned with the manner in which such a concept is superimposed to yield meaningful results.

Abstract Objects

We need the following symbolism and terminology.

$\mathscr{A}$: A finite set of "elementary decisions" a, b, c,..., sometimes referred to as the "alphabet" or "letters."

X: A set of "policies," or "words," of *finite* length, formed by the concatenation of elementary decisions: $\underline{x} \in X = \{\underline{x} : \underline{x} = a_1 a_2 a_3 \ldots a_m\}$, $a_i \in A$. The length of $\underline{x}$ is denoted by $L(\underline{x}) = m$, finite. X is also interpreted as the "space of policies" to a (hypothetical) oriented process or "black box."[2] A policy $\underline{x} \in X$ may be a vector and may be a function of time, i.e., varies with time, $\underline{x} = \underline{x}(t)$, $t \in T$. In general, T may stand for any indexing set (oftentimes $t \in T$ indicates the *stage* or *epoch*, to be defined below).

Concatenation, Continuation, and Prefix.

If $\underline{x}_1$ and $\underline{x}_2$ are policies in X, of length m_1 and m_2, resp. then the *concatenation* of $\underline{x}_1$ and $\underline{x}_2$ is a new policy $\underline{x} = \underline{x}_1\underline{x}_2 \in X$, and $L(\underline{x}) = m_1 + m_2$. The operation of concatenation permitted us to construct the individual policies, say $\underline{x}_1$, in the first place from the set of elementary decisions $\mathscr{A}$. Obviously, X is associative, i.e.,

$$\underline{x}_1(\underline{x}_2\underline{x}_3) \equiv (\underline{x}_1\underline{x}_2)\,\underline{x}_3\,, \qquad \underline{x}_1\,, \underline{x}_2\,, \underline{x}_3 \in X.$$

Furthermore, X is closed under concatenation, i.e.,

$$\text{if } \underline{x}_1 \in X,\ \underline{x}_2 \in X \Rightarrow \underline{x}_1\underline{x}_2 \triangleq \underline{x} \in X.$$

A policy $\underline{x} \in X$ is said to have a *prefix* $\underline{x}_1 \in X$ if there exists a policy $\underline{x}_2 \in X$ s.t. $\underline{x} = \underline{x}_1\underline{x}_2$. And in this case $\underline{x}_2$ is called the *continuation* of $\underline{x}_1$ under $\underline{x}$ (or to form $\underline{x}$).

[2] We are forced to use these words before giving their precise definitions (see below). For the time being, they can be understood to stand for their common interpretation.

V: Space of output value of the (hypothetical) oriented object or "black box". We assume that V is a linear, completely ordered metric space (such as the real line), and if the input policy $\underline{x}$ varies with time v may also vary with time

DEFINITION. *Non-Oriented Abstract Objects.*[3] Almost by definition, Operations Research problems lend themselves naturally to representation by *oriented objects*, i.e., objects whose inputs and outputs are defined. This is due mainly to the optimizing nature of such problems. However, it is helpful to extract oneself from such conceptual confinement and visualize "background" problems which, as originally stated, are free of any orientation. Then we may speak of non-oriented objects. Thereafter, we may impose an orientation on them to suit not only our convenience but also some practical application. Still, we may retain the privilege of changing the orientation whenever that is to our advantage — either conceptually or practically. An abstract object defines, as well as is defined by, a family of input output relation $(\underline{x}, \underline{v})$.

The majority of physical laws fall in the category of models of non-oriented objects. For instance, consider an inventory system which has three attributes: the amount ordered and received at the beginning of period t, a_t; the quantity sold during the period, d_t; and the residual quantity at the end of the period (i.e., the end-of-period inventory, which may be positive or negative), i_t. The relation between these attributes is given by

$$i_t - i_{t-1} - a_t + d_t = 0; \qquad t = 1, 2, \ldots, T \tag{3}$$

with i_0 a known constant. This relation *defines* a *non-oriented* object Θ, because we have not specified what is to be taken as "input" and what is to be taken as "output."

Suppose that now we impose such an orientation to obtain the *oriented* object $\mathcal{O}$: let a_t be the "input" (or elementary decision) at period t; $t = 1, 2, \ldots, T$; and let i_t be the output. Since the vector of demand $d = (d_t)$ is assumed a known vector, we immediately have the output in terms of the input, the so-called *input-output relation* of $\mathcal{O}$:

$$i_t - i_{t-1} = a_t - d_t . \tag{4}$$

Here, given the purchase quantities (a_t) as input we deduce the end-of-period inventories (i_t) as output.

[3] The adjective "abstract" is to free the discussion from questions of physical realizability. We wish to feel free to discuss processes characterized by relations containing the imaginary number $j = \sqrt{-1}$, such as $v = jx$. To every physical (i.e., realizable) object there corresponds an abstract process, but not conversely.

It is important to remark that this orientation is completely arbitrary. For instance, we could have chosen (i_t) as the input to (another) oriented object $\mathscr{B}$, and defined a_t as the output of $\mathscr{B}$ to obtain the input-output relation of $\mathscr{B}$:

$$a_t = i_t - i_{t-1} + d_t \tag{5}$$

Here, given the end-of-period inventories (i_t), $t = 0, 1, 2, \ldots, T$, as input we can deduce the purchase quantities (a_t) as outputs.

DEFINITION. *Oriented Object* $\mathscr{O}$: *Discrete Decision Process* (DDP). The above discussion leads immediately to defining an *oriented* object to be a family of ordered pairs $(\underline{x}, \underline{v})$ in which the first component $\underline{x}$ is called the *input* (or *policy*) and the second component $\underline{v}$ is called the *output* (or, response, or, if measured in economic terms, the *payoff* or *value*). A pair $(\underline{x}, \underline{v})$, which may or may not be time functions, is an *input-output-pair belonging to* $\mathscr{O}$, written $(\underline{x}, \underline{v}) \in \mathscr{O}$ iff $\underline{x} \in X$ and $\underline{v} \in V$.

Thus an (abstract) oriented object $\mathscr{O}$, or a DDP, is *defined* in terms of the totality of its input-output pairs, *and conversely*, an abstract oriented object $\mathscr{O}$ *defines* policies $\underline{x} \in X$ and outputs $\underline{v} \in V$ such that pairs $(\underline{x}, \underline{v})$ are input-output pairs to $\mathscr{O}$.

Notice that the manner in which an abstract oriented object is defined is similar to the definition of a *relation* (rather than a function, an operator, or a mapping) in mathematics. As is well known, a "relation" is simply a set of ordered pairs, while a function (or mapping) is a relation in which to every $\underline{x}$ there corresponds a unique $\underline{v}$. We shall indicate such a relation by

$$\mathscr{O}(\underline{x}, \underline{v}) = 0, \tag{6}$$

the *input-output relation* of $\mathscr{O}$, which defines the set of all input-output pairs belonging to $\mathscr{O}$.

If we denote the range of $\underline{x}$ by $\mathscr{R}(\underline{x})$, and the range of $\underline{v}$ by $\mathscr{R}(\underline{v})$, we remark that $\mathscr{O}$ is a subset of the space formed by the Cartesian product $\mathscr{R}(\underline{x}) \times \mathscr{R}(\underline{v})$.

An excellent example of an input-output relation is Eq. (3) after specifying the inputs and outputs, as was done in (4) or (5), with $\mathscr{R}(i_t) = (-\infty, +\infty)$ and $\mathscr{R}(a_t) = [0, +\infty)$.

DEFINITION. *The Concept of Stage* (*or Epoch*). Because of the discrete nature of the input (i.e., policy), we define a *stage*, or *epoch*, to be the index corresponding to an elementary decision. In other words, if

$$\underline{x} = a_1 a_2 \cdots a_i a_{i+1} \cdots a_m,$$

then we take the sentence "the object $\mathscr{O}$ is at stage i" to mean that (elementary) decisions $a_1 \cdots a_{i-1}$ have been made and the object is ready for the input

decision a_i. And conversely, the elementary decision a_i (i.e., input a_i) implies the existence of a stage i in which the decision takes place.

In many cases, the stage (or epoch) coincides with a discrete time interval. For instance, in the Dynamic Economic Manufacturing Quantity Problem (Example 3), we speak of the stages to mean the periods (the days, weeks, or months, etc.) in which decisions are made. However, the notions of stage and time intervals do not coincide in the other three examples.

Thus the notion of stage is *independent of time.* As defined above, a stage may exist even though the policy $\underline{x}$ is determined in its entirety at the same instant of time. According to our definition, the number of stages (or epochs) is related to the length of the policy $\underline{x}$, $L(\underline{x})$, and not to the real time in which the policy (or elementary decisions) takes place.

Input (i.e., Decision) and Output (i.e., Value) Segments

We shall sometimes find it more convenient to speak of the *input segment* $\underline{x} = a_1 a_2 \cdots a_m$ in its totality, as well as of the corresponding *output segment* $\underline{v} = b_1 b_2 \cdots b_m$, instead of individual (i.e., elementary) decisions a_i and individual output values b_i. This is a slight departure from the common practice of speaking of "the output" to mean b_m, which is the *terminal* value of the output segment at the end of stage m. Whenever we wish to refer to the value b_m we shall always speak of the *terminal output* b_m or, alternatively, of *the output at stage m.*

In other words, the definition of $\underline{x}$ as a policy (or word) formed by the concatenation of elementary decisions (or alphabet $a_1 a_2 \cdots a_i a_{i+1} \cdots a_n$) permits one to deduce, by induction on $L(\underline{x})$, that

$$\underline{x}_1 \triangleq a_1 a_2 \cdots a_m$$

is a *prefix* to

$$\underline{x}_2 \triangleq a_{m+1} a_{m+2} \cdots a_n$$

such that

$$\underline{x} = \underline{x}_1 \underline{x}_2 .$$

Conversely, $\underline{x}_2$ is a *continuation of* $\underline{x}_1$ *in* $\underline{x}$ such that $\underline{x} = \underline{x}_1 \underline{x}_2$. It is sometimes convenient to talk about a segment $\underline{x}'$ of $\underline{x}$ to mean, say, that

$$\underline{x} = a_1 a_2 \cdots a_{i-1} \, \underline{x}' \, a_r \cdots a_n$$

where $\underline{x}' \equiv a_i a_{i+1} \cdots a_{r-1}$ is a sub-policy which is a continuation of the sub-policy $a_1 a_2 \cdots a_{i-1}$ in $\underline{x}$ such that $a_1 a_2 \cdots a_{i-1} \, \underline{x}'$ is a prefix to the sub-policy $a_r a_{r+1} \cdots a_n$ in $\underline{x}$.

Similar definitions apply to the output (i.e., response value) segment $\underline{v}$. We shall oftentimes talk of the response segment $\underline{v}'$ corresponding to the input segment $\underline{x}'$ to imply the following: $\underline{x}'$ is a segment of an input $\underline{x}$, in the sense

defined above. $\underline{v}$ is an output of $\mathcal{A}$ corresponding to $\underline{x}$ in the pair $(\underline{x}, \underline{v}) \in \mathcal{A}$; and $\underline{v}'$ is a segment of $\underline{v}$ corresponding to $\underline{x}'$ in the pair $(\underline{x}', \underline{v}') \in \mathcal{A}$. Notice that such a segment $\underline{v}'$ must exist; the proof of its existence follows from the definition of $\mathcal{A}$. That definition, in order to be completely consistent must also stipulate that $\mathcal{A}$ includes all pairs $(\underline{x}', \underline{v}')$, where $\underline{x}'$ is a segment of $\underline{x}$ and $\underline{v}'$ is a segment of $\underline{v}$. Notice, again, that the space of $\mathcal{A}$ is contained in the space formed by the Cartesian product of the range of segments $\underline{x}'$ and the range of segments $\underline{v}'$.

In other words, the input-output response segments are the set of ordered pairs $\underline{x}'$, $\underline{v}'$, where $(\underline{x}')$ is the set of input segments of $\underline{x}$ defined by

$$\underline{x}_1' = a_1 ; \qquad \underline{x}_2' = a_1 a_2 ; \qquad \cdots ; \qquad \underline{x}_n' = a_1 a_2 \cdots a_n = \underline{x}$$

and $(\underline{v}')$ is the set of output segments of $\underline{v}$ defined by:

$$\underline{v}_1' = b_1 ; \qquad \underline{v}_2' = b_1 b_2 ; \qquad \cdots ; \qquad \underline{v}_n' = b_1 b_2 \cdots b_n = \underline{v} .$$

In a sense, a segment (of either $\underline{x}$ or $\underline{v}$) is similar to the complete curve of a time function, rather than its value at a particular time t.

The Concept of State

It is important to recognize the rather elementary, i.e., highly *general*, structure with which we are defining a DDP. We assume given the input space X, the output space V, and the input-output relation $\mathcal{A}(\underline{x}, \underline{v}) = 0$.

We have introduced the concept of stage in terms of the length of the input $\underline{x}$. We now introduce the concept of state in terms of these concepts.

As was remarked before, the input-output relation (6) is *not* a function (i.e., mapping) from $\underline{x}$ into $\underline{v}$ (or from $\underline{v}$ into $\underline{x}$). In fact, for the same input $\underline{x}$ (output $\underline{v}$) there may correspond several outputs (inputs) depending on the (still vague) notion of "*state*" of $\mathcal{A}$. The notion of "state" will be made precise below, but it is clear, at this juncture, that the "state" must be a device to parametrize the space of input-output pairs. In other words, the "state" of $\mathcal{A}$ is a means to achieve the representation of $\mathcal{A}$ as a mapping (usually from x into v).

For example, it is clear that the value of a tour in the Traveling Salesman Problem depends on his starting city. In other words, for any feasible π in the space of all feasible tours, the total cost $v(\pi)$ of Eq. (1) is not uniquely determined, though we may be able to write down the input-output pairs.

Suppose we denote the state of $\mathcal{A}$ by s, and the space of all possible states by S, and the range of s by $\mathcal{R}(s)$. As was pointed out above, s is a label attached to every (x, v)-pair such that v is uniquely determined by x and s. *This defines the state as a parametrization of the input-output pairs* (x, v).

A basic question is the following: given an oriented object $\mathcal{O}$ (i.e., a discrete decision problem; DDP) defined by its input-output relations (6), how can one associate a state space S to $\mathcal{O}$? Certainly, a clue to the answer to this question must be given by the answer to the following secondary question: what has one to know about $\mathcal{O}$ (e.g., its "history" prior to the application of $\underline{x}$) that would, together with $\underline{x}$, define $\underline{v}$ uniquely? Interestingly enough, the answer to this question, though necessary to the construction of S, is not sufficient because other conditions, the so-called "consistency conditions" and the "decomposition" property, must be imposed for the resulting state to be of full operational value.

Thus the construction of S falls into two steps: In the first we stipulate the existence of several spaces (S) that are candidates for the parametrization of $\mathcal{O}$. (Of course, if no such S exists, then there is no point in pursuing the subject any further.) Then we impose the consistency and decomposition conditions on S which would eliminate some candidates and leave (hopefully) at least one candidate state space S.

The consistency conditions are themselves properties, or axioms, which we expect any S to possess (similar to the more familiar axioms of linear vector spaces, groups, rings, etc., in algebraic structures). The decomposition condition is a property of the state that permits its representation as a vector one component of which is the value of "being" in that state.

Stipulation. We stipulate that the spaces of input-output pairs of $\mathcal{O}$ admit parametrization in the form of a functional relation

$$v = G(s, x), \tag{7}$$

where G denotes a single-valued function (i.e., mapping) of s and x into v; s ranges over S, x ranges over X and v ranges over V.

If Eq. (7) satisfies the mutual- and self-consistency conditions as well as the decomposition property, all of which are cited below, then S is called the *state-space* of $\mathcal{O}$, $s \in S$ is the initial state of $\mathcal{O}$; and Eq. (7) is called the *input-output-state equation* of $\mathcal{O}$. Furthermore, as we shall see below, we shall say that:

$\mathcal{O}$ is completely characterized by (7); as it was characterized by (6);

v is the output (or response) of $\mathcal{O}$ to the input (i.e., policy) x starting in state s;

x and v constitute an *input-output-pair with respect to* s.

An object $\mathcal{O}$ which is defined in terms of its input-output-state equation will be sometimes referred to as a "*sequential discrete decision process*," SDDP.

Notice that Eq. (7) gives the value of v as the *result* of the object $\mathcal{O}$ starting

in state s and being subjected to the input (i.e., policy) x. In other words, v is represented in (7) as an ordinary point function.

As we have remarked above, it is sometimes necessary to talk about the complete response (i.e., output) segment $\underline{v}$, and for ease of notation we write

$$\underline{v} \triangleq \bar{G}(s, \underline{x}) \triangleq b_1 b_2 \cdots b_n . \tag{8}$$

The difference between the mappings G and $\bar{G}$ is similar to the difference between the value of a function *at* a certain point of time t and the complete curve between the starting time t_0 and t.

The Mutual- and Self-Consistency Conditions on S

We now state the mutual-and self-consistency conditions which must be satisfied by a candidate S to be eligible as a "state space."

Condition I. *The Mutual Consistency Condition.* Every input-output pair (x, v) of $\mathcal{O}$ satisfying (6) also satisfies the input-output state Equation (7), for some $s \in S$, *and vice versa.*

The condition is not a trivial one, since it is relatively easy to find examples of DDP's and candidates S which do not satisfy it. But before giving such an example, we rephrase the Condition to read as follows: Suppose the pair (x, v) satisfies (6); then we require that there exists an $s \in S$ such that $v = G(s, x)$ *and conversely*, any pair x and v satisfying (7) for some $s \in S$ is also an input-output pair for $\mathcal{O}$; i.e., they also satisfy (6). In essence, Condition I guarantees that (6) and (7) represent the same object $\mathcal{O}$.

As an example of a state space which does *not* satisfy Condition I, let $\mathcal{O}$ be defined by the input-output relation

$$\mathcal{O}(x, v) = v_n - v_{n-1} - a_n = 0 \qquad \text{for} \quad n = 1, 2, 3, \ldots, \tag{9}$$

where $a_n \in \mathcal{A}$. Now consider the input-output-state equation

$$v_n = \beta + \sum_{i=1}^{n} a_i , \qquad \beta \geqslant 1. \tag{10}$$

Seemingly, β is a scalar which purports to represent the "state" of $\mathcal{O}$. In this case, suppose we restrict

$$\mathcal{R}(\beta) \quad \text{to} \quad \mathcal{R}(\beta) = [1, +\infty), \quad \text{while} \quad \mathcal{R}(x) = \mathcal{R}(v) = (-\infty, +\infty).$$

It is clear that any v_n, β and (a_i) satisfying (10) also satisfy (9), but *not* conversely, since there is no point in $\mathcal{R}(\beta)$ which satisfies the input-output

pairs $(v_1 = 1, a_1 = 1)$, $(v_2 = 2, a_2 = 1)$, $(v_3 = 3, a_3 = 1)$, $(v_n = 3, a_n = 0$ for $n \geqslant 4)$, since

$$v_3 = \beta + \sum_{i=1}^{3} a_i = \beta + 3 = 3 \Rightarrow \beta = 0 \notin \mathscr{R}(\beta).$$

In spite of our assertion of the nontriviality of this condition, we hasten to add that it is not particularly difficult to satisfy it by the proper enrichment of the state space S.

CONDITION II. *First Self-Consistency Condition.* The response v is uniquely determined by s and x.

In order for any $s \in S$ to qualify as a state, S must have the property that, given any $s \in S$ and any input (i.e., policy) $x \in X$ *which is permissible at* s (see below) the output of $\mathscr{O}$ is uniquely determined and hence is *independent of any other information.*

This is a key property in the notion of state. Condition II is a precise formalization of the notion of the "regenerative character" of the state alluded to by several writers on dynamic programming.

It is important to realize that Condition II does *not* imply that $\mathscr{O}$ is of zero memory. This would certainly be a stronger condition, *which is not needed.* Condition II simply emphasizes that *no information other than* s *and* x *is needed* for the evaluation of v. Indeed, in some instances, the state s itself may provide some of the history of $\mathscr{O}$.

For instance, consider Example 1, the Traveling Salesman Problem, and Example 2, the "Faithful" Traveling Salesman Problem. Suppose we define the state to be a vector of three numbers $(m, \underline{k}, \xi)$, where m is the number of cities visited by the Salesman outside his base city (which we assume, for the moment, to be city $\underline{1}$), $\underline{k}$ is the last city visited and ξ is the distance covered thus far. It is not difficult to see that the space S of all feasible states is given by the Cartesian product of $m \in \{2, 3, \ldots, n\} \times k \in \{2, 3, \ldots, n\} \times \xi \in [0, +\infty)$. Furthermore, it is easy to show that defining the state in this fashion satisfies Condition II relative to the Traveling Salesman *but not relative to the "Faithful" Traveling Salesman Problem.* The reason is that knowing

$$m \in \{2, 3, \ldots, n\}, k \in \{2, 3, \ldots, n\} \qquad \text{and} \qquad \xi \in [0, +\infty)$$

is not sufficient to determine the cost $v(\pi)$ of Eq. (2) uniquely for any input x, since if the Salesman has been away from home in his last trip r days, then an input x whose $L(x) > 5 - r$ causes $v(\pi)$ to be equal to $+\infty$, while $v(\pi)$ has a finite value otherwise'

It is obvious, therefore, that in the case of the "Faithful" Traveling Salesman, we must add such information to the definition of this "state." This

results in a vector of *four* entries: $(m, \underline{k}, r, \xi)$, where m and $\underline{k}$ are as defined before, ξ is now the *cost* incurred thus far, and r is the number of cities visited after the last stay in city $\underline{1}$. Notice that we require from the state to "remember" not only the number of cities visited, but also the total cost incurred to date, the last city visited as well as the number of cities visited since his last visit to city $\underline{1}$! Certainly this is no "memoryless" object.

DEFINITION. *"Permissible" Policies at a State.* Before continuing with the discussion of the consistency conditions, we dwell for a moment on the concept of permissible (elementary) decisions and permissible policies at a state $\underline{s}$.

We recall that a state was derived as a parametrization of input-output pairs $\{(x, v)\}$. That is, one can visualize starting with a long "list" of (x, v) pairs, to which we have added states to distinguish among identical inputs to $\mathcal{A}$ which result in different outputs.

After completing this process, one can certainly visualize the rewriting of the list according to the newly derived states. In other words, by indexing the states in an arbitrary fashion as 1, 2,..., S, one can group together all input-output pairs which are common to state s, $s = 1, 2, \ldots, S$. We now *define* the *permissible policies at state s as the totality of elementary decisions and policies which appear in the set of input-output pairs appearing in the list as "belonging" to state s.* Clearly, they are a subset of all possible policies, and shall be designated by $X_s \subset X$.

We now give

CONDITION III. *The Second Self-Consistency Condition.* Let (x, v) be any input-output pair satisfying (6) and (7) for some state $s_0 \in S$, i.e., $\underline{v} = \bar{G}(s_0, \underline{x})$. Let $\underline{x}_1$ be the generic designation of the first segment of $\underline{x}$ for all such possible segments, and let $\underline{v}_1$ be the corresponding first segment of $\underline{v}$, as defined above. Let $\underline{x}_2$ be the continuation segment of $\underline{x}_1$ in $\underline{x}$, and $\underline{v}_2$ the corresponding continuation segment of $\underline{v}_1$ in $\underline{v}$. Then we require that for every possible choice of $(\underline{x}_1, \underline{v}_1)$, and for every feasible continuation of $\underline{x}_1$ (in all permissible policies $\underline{x}$ whose first segment is $\underline{x}_1$), there exists at least one state $s \in S$ for which the output $\underline{v}_2$ is related to the continuation input $\underline{x}_2$ by

$$\underline{v}_2 = \bar{G}(s; \underline{x}_2). \tag{11}$$

We emphasize that the state s does not depend on the continuations $\underline{x}_2$.

To gain insight into this condition, which is of central importance in our development, fix the segment $\underline{x}_1$ and let $Q(s_0; \underline{x}_1 \underline{x}_2)$ be the set of all states in S with respect to which (x_2, v_2) is an input-output pair satisfying (8); i.e.,

$$Q(s_0; \underline{x}_1\underline{x}_2) \triangleq \{s \in S \mid \underline{v}_2 = \bar{G}(s, \underline{x}_2) \quad \text{and} \quad \underline{v}_1\underline{v}_2 = \bar{G}(s_0; \underline{x}_1\underline{x}_2)\}. \tag{12}$$

Condition III requires that as $\underline{x}_2$ is varied over all possible continuations of $\underline{x}_1$ in the space of feasible policies X, the set $Q(s_0 ; \underline{x}_1\underline{x}_2)$ also varies but *the intersection*

$$Q(s_0 , \underline{x}_1) \triangleq \bigcap_{\underline{x}_2} Q(s_0 ; \underline{x}_1\underline{x}_2) \tag{13}$$

is non-empty (see Fig. 1).

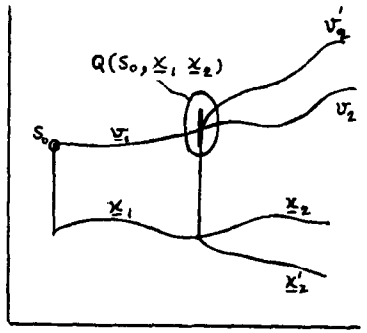

FIGURE 1

Consider any input segment $\underline{x}_1\underline{x}_2$ and a corresponding output segment $\underline{y}_1\underline{y}_2$ such that $\underline{y}_1\underline{y}_2 = \bar{G}(s_0 ; \underline{x}_1\underline{x}_2)$. To say that $(\underline{x}_2 ; \underline{y}_2)$ also satisfy (8) is to say that there exists a state $s \in S$ such that $\underline{y}_2 = \bar{G}(s, \underline{x}_2)$. The set of all such states is the set $Q(s_0 ; \underline{x}_1\underline{x}_2)$. If the intersection in (13) is non-empty, i.e., if Condition III is satisfied, then it must be true that any individual $Q(s_0 ; \underline{x}_1\underline{x}_2)$ is itself non-empty. Therefore, in some sense, the first implication of Condition III is that the state space S is sufficiently "rich" to include all possible initial states of $\mathcal{A}$.

The following question may linger in the reader's mind: is it possible for $Q(s_0 ; \underline{x}_1\underline{x}_2)$ to be empty? In other words, is it possible that there exists an object $\mathcal{A}$ with state space S and initial state $s_0 \in S$; input $\underline{x}_1\underline{x}_2$ and output $\underline{y}_1\underline{y}_2$ satisfying (8) but there exists no state $s \in S$ for which the triplet $\underline{x}_2 , \underline{y}_2 , s$ satisfies (8)? Remember that we are asking this question relative to an *individual* input-output pair $(\underline{x}_1\underline{x}_2 ; \underline{y}_1\underline{y}_2)$, which is a less demanding condition than the intersection of (13).

The answer, surprisingly enough, is yes. As the following example demonstrates, it is possible to define a state space S, an initial state s_0, an input $\underline{x}_1\underline{x}_2$, and a corresponding output $\underline{y}_1\underline{y}_2 = \bar{G}(s_0 , \underline{x}_1\underline{x}_2)$ but $Q(s_0 ; \underline{x}_1) = \phi$.

Consider the example of Eqs. (9) and (10) but with the state space $S = \{0\}$; i.e., the space contains a singleton point $s = 0$. Let $V \triangleq \{v_n \mid v_n = n; n = 0, 1, 2,...\}$ and $X \triangleq \{\underline{x}_n \mid \underline{x}_n = a_1a_2 \cdots a_n ; a_i = 1 \forall i \leqslant n; n = 1, 2,...\}$.

Then it is easy to see that Condition I is satisfied, because any pair of input-output segments $(\underline{x}, \underline{v})$ satisfying

$$v_n - v_{n-1} = a_n$$

also satisfies

$$v_n = 0 + \sum_{i=1}^{n} a_i ,$$

where the zero is the initial state of the system. And conversely, any triplet $(0; \underline{v}_n \in V; \underline{x}_n \in X)$ which satisfies the second equation also satisfies the first. Furthermore, the function

$$v_n = G(s; \underline{x}_n) = 0 + \sum_{i=1}^{n} a_i$$

determines v_n uniquely for the given s and any $x_n \in X$. Hence, Condition II is also satisfied.

However, consider the output segment $\underline{v} = \bar{G}(s, \underline{x})$ starting at $s = 0$ with $\underline{x}_n = a_1 a_2 \cdots a_n$. Clearly, $v_1 = 1$; $v_2 = 2$, $v_3 = 3$; $v_n = n$. Now write $\underline{x}_n = \underline{x}_1 \underline{x}_2 = (a_1 a_2 \cdots a_m) \circ (a_{m+1} a_{m+2} \cdots a_n)$; write

$$\underline{v}_n = \underline{v}_1 \underline{v}_2 = (v_1 v_2 \cdots v_m) \circ (v_{m+1} v_{m+2} \cdots v_n)$$

in which $\circ$ denotes the concatenation of the two segments. Obviously

$$\underline{v}_1 \underline{v}_2 = \bar{G}(0; \underline{x}_1 \underline{x}_2) = \Big\{0 + \Big(\sum_{j=1}^{k} a_j\Big)\Big|_{k=1}^{n} .$$

However, there exists no $s \in S$ satisfying the relation

$$v_2 = G(s, \underline{x}_2) = s + \sum_{i=j+1}^{n} a_i ; \qquad v_2 \in V; \qquad \underline{x}_2 \in X.$$

For instance,

$$v_{j+1} = j + 1 = s + a_{j+1} = s + 1 \Rightarrow s = j \notin S.$$

Returning now to the intersection of (13), we immediately see that the requirement that it be non-empty makes it both possible and meaningful for us to speak of *the state s_m of $\mathcal{A}$ at the termination of the input segment $\underline{x}_1 = a_1 a_2 \cdots a_m$ when $\mathcal{A}$ starts in state s_0*, in which s is independent of any input continuation segment $\underline{x}_2$ and the corresponding output continuation $\underline{v}_2$. Otherwise, such a statement would be meaningless, for there would exist no state in S which can account for all possible input-output continuations $(\underline{x}_2, \underline{v}_2) \in X \times V$.

Remark 1. As a remark on the development thus far, it is obvious that if we establish a state space S which satisfies all three consistency conditions, then any relabeling of the states (or, in a more formal language, any one-one mapping of S onto itself) cannot cause S to violate any of the three conditions. Sometimes such one-one mapping is advisable for computing and other purposes, and in many cases it amounts to a mere change in the coordinates of S.

The Decomposition Property of the States

Condition IV. The state s is a vector of the form

$$s \equiv (v, \alpha)$$

where v is the value of the output of $\mathcal{A}$ immediately preceding the application of the input x and α is a (possibly null) vector of all other parameters unrelated to v. (If α is the null vector, as distinguished from the zero vector, it is not written at all in the expression of s.)

It is obvious that if $\mathcal{A}$ is in its initial state and no input has been applied yet, one can then speak of v_0 as the initial (i.e., zeroth) value of the output. On the other hand, it is not difficult to show, and we shall do that below, that the state s_m at the mth stage is a function of the terminal value v_m and the input x. Condition IV emphasizes the separability of v and α.

In some sense, Condition IV and the decomposition property it specifies are "natural" requirements when viewed in the light of Condition III. After all, if the intersection in (13) is non-empty and the output segment $\underline{v}_2$ can be represented in terms of s and $\underline{x}_2$ *only*, it is meaningful to require that the state s contain all the information that is needed for the SDDP to "pick up" from state s and evaluate the outputs in the subsequent stages $m + 1, m + 2, \ldots, n$.

The State Equation

As indicated above, if the input policy $\underline{x}$ is of length $L(\underline{x}) = m$, we refer to the state $s_m \in Q_m(s_0, x)$ as the "state of $\mathcal{A}$ at the end of stage m", or simply "the state of $\mathcal{A}$ at stage m."

In some instances, such as in proving certain properties of the state space S, it is helpful to exhibit the exact transformation occurring in the state of $\mathcal{A}$ when it starts initially in state s_0 and input (i.e., policy) $\underline{x} = a_1 a_2 \cdots a_m$ is applied to it. Thanks to Condition III, it is now meaningful for us to speak of the "state of $\mathcal{A}$ at stage m", $s_m \in Q_m(s_0, x)$.

Recall that, originally, we are given the object $\mathcal{O}$, which is synonymous with saying that we were given the input-output relation

$$\mathcal{O}(\underline{x}, \underline{v}) = 0.$$

From the input-output relation we derived the parametrization $\{s\} \triangleq S$, which we called the "state space" of $\mathcal{O}$ and which satisfied three consistency conditions and a decomposition property and resulted in the functional relation

$$v = G(s_0, x)$$

or

$$\underline{v} = \bar{G}(s_0, \underline{x}),$$

which we called the *input-output-state equation.*

Consequently, the state s_0 is uniquely determined (to within equivalent classes, which will be explained below) by the input-output pair (x, v); i.e.,

$$s_0 = g(x, v). \tag{14}$$

Such representation is possible by the very definition of the state s in terms of the input-output pairs and the derivation of the input-output-state equation.

Now consider any $s \in Q(s_0, x)$. By Condition III, Q is not empty and is independent of any continuation of x. Hence s is dependent only on s_0 and x, say

$$s_m = W(s_0, x). \tag{15}$$

This is the *state equation* of $\mathcal{O}$. It defines the state s_m at stage m as a function (i.e., single-valued mapping) of the initial state of the object, s_0, and the input $\underline{x} = a_1 a_2 \cdots a_m$. For brevity of notation, we shall also write

$$\underline{s} = \bar{W}(s_0, \underline{x}) \tag{16}$$

whenever we wish to indicate the value of the "state segment" over the m stages defined by $\underline{x}$. In other words $W(s_0, \underline{x})$ defines all pairs of stage and state, (i, s_i), under the input $\underline{x} = a_1 a_2 \cdots a_m$.

Notice that, by the decomposition property

$$\begin{aligned} s_m &\equiv (v_m, \alpha_m) = (G(s_0, x); \alpha_m) = [G((v_0, \alpha_0), x); \alpha_m] \\ &= W(s_0; x) = W((v_0, \alpha_0); x). \end{aligned}$$

Since $v_m = G((v_0, \alpha_0); x)$, it must be true that the transformation W on the pair $(s_0; x)$ is separable into two transformations

$$W_1[(v_0, \alpha_0); x] = v_m = G[(v_0, \alpha_0); x] = G(s_0; x);$$

and

$$T(\alpha_0, x) = \alpha_m. \tag{17}$$

It is this latter transformation T, *derived* above from the basic input-output pairs and Conditions I to IV, that is commonly denoted by "the state transformation *function*" and is assumed *given* in previous work. Notice that, in contradistinction from other work in dynamic programming (see, e.g., Denardo and Mitten [4]), we insist that W, W_1 and T be single-valued functions of their arguments. We contend that, with the proper definition of the state space, this assumption involves no loss of generality whatsoever.

Again, if we wish to denote the complete segment of the α-portion of the states under the input x when $\mathcal{O}$ starts initially in state $s_0 = (v_0, \alpha_0)$, we shall write

$$\underline{\alpha} = \bar{T}(\alpha_0, \underline{x}) \triangleq T(\alpha_0, a_1) \circ T(\alpha_1, a_2) \circ T(\alpha_2, a_3) \circ \cdots \circ T(\alpha_{m-1}, a_m) \quad (18)$$

in which the $\circ$ denotes the concatenation of the elements. And for the sake of economy of expression, we shall abbreviate the "α-portion of a state" to the "α-state," with the understanding that a "state" without any prefix refers always to $s \equiv (v, \alpha)$.

Remark 2. It can be easily seen that Condition IV is neither trivial nor derivable from the other three conditions, i.e., it is an independent condition. In fact, from Condition III we know that

$$s_m = W(s_0 ; x_1) \in Q(s_0, x_1), \qquad x_1 \equiv a_1 a_2 \cdots a_m$$

and

$$s_0 = g(v_m, x_1)$$

by the very process of constructing the state space S from the input-output pairs $\{(x, v)\}$. Thus we immediately have

$$s_m = W(g(v_m, x_1); x_1).$$

This leads to the conclusion that s_m is indeed dependent on v_m and x_1. Since, by hypothesis, S is a vector space, each $s \in S$ is a vector of dimensionality k, say. We are now able to conclude that each component of that vector is a *function* of $\underline{v}_m$ and $\underline{x}_1$ only.

Condition IV asserts much more than that: it asserts that v_m itself is a component of the vector, and that no other component of s_m is a function of v_m ; i.e., the α-component of s_m is independent of v_m .

THEOREM 1 (*The Output Separation Property* (*Output-SP*)). *For any input segment* $\underline{x}_1 = a_1 a_2 \cdots a_m$ *followed by a feasible continuation segment* $\underline{x}_2 = a_{m+1} a_{m+2} \cdots a_n$, *the response of* $\mathcal{O}$ *to* $\underline{x}_1 \underline{x}_2$ *starting in state* s_0 *consists of the response segment* $\bar{G}(s_0, \underline{x}_1)$ *followed by the response segment* $\bar{G}(s_m, \underline{x}_2)$, where s_m *is the state of* $\mathcal{O}$ *at the end of stage* m.

Proof. The proof follows immediately from Eq. (12). By definition of s_m ,

see Eq. (13), it is an element of $Q_m(s_0, x_1)$ which, from (12) is the set of all states satisfying

$$\bar{G}(s_0, \underline{x}_1\underline{x}_2) = \bar{G}(s_0, \underline{x}_1) \circ \bar{G}(s_m, \underline{x}_2) \triangleq \underline{v}_1\underline{v}_2, \forall s_0, \forall \underline{x}_1, \underline{x}_2 \in X, \quad (19)$$

in which $\bar{G}(s_0, \underline{x}_1) \circ \bar{G}(s_m, \underline{x}_2)$ denotes the concatenation of the two output segments. In particular, the value of the output at the end of stage n is given by

$$\begin{aligned} v_n = G(s_0; x_1x_2) &= G(s_m; x_2), \qquad m = 1, 2, \ldots, n, \\ &= G[W(s_0, x_1); x_2], \end{aligned} \quad (20)$$

or, in words, the value of the output at the end of stage n is equal to the output when $\mathcal{A}$ starts in state s_m and receives the input $x_2 = a_{m+1}a_{m+2} \cdots a_n$.

Remark 3. The output-SP gives another precise definition of the notion of the state as a "point of regeneration" of $\mathcal{A}$, referred to by several authors.

Properties of the State Space

Definition. *"Reachable" States and "Descendant" States.* A state $s' \in S$ is *reachable from a state* $s \in S$ iff there exists an input segment $\underline{x} \in X$ of *finite length* (i.e., $L(x) < +\infty$) such that s is transformed (under the mapping W) into s' by x; i.e., iff $s' = W(s, x)$ for some $\underline{x} \in X$ and $L(x) < +\infty$.

We shall refer to the set of all states reachable from s (for all finite inputs $x \in X$) as the "*descendant states of* s," or simply the "descendants of s," denoted by S_s.

Perhaps one of the most important concepts in systems analysis is that of "equivalent states." For our purposes, we need the

Definition. *Equivalent States.* Two states $s_1 \equiv (v_1, \alpha_1)$ and $s_2 \equiv (v_2, \alpha_2)$ are said to be equivalent iff $\bar{T}(\alpha_1, x) \equiv \bar{T}(\alpha_2, x)$ for all $x \in X$; i.e., iff the descendant α states are identical under all permissible policies $x \in X$.

State equivalence is indicated symbolically by $\simeq$; i.e.,

$$s_1 \simeq s_2 \Leftrightarrow \bar{T}(\alpha_1, x) \equiv \bar{T}(\alpha_2, x) \; \forall \, x \in X. \quad (21)$$

Two states s_1 and s_2 are *distinct* if their vector representations are not equal in the ordinary equality-of-vectors sense.

Two states s_1 and s_2 are non-equivalent, or, better still, distinguishable, if they are not equivalent in the sense of the above definition (21). More formally,

Definition. *Distinguishable States.* Two states s_1 and s_2 are distinguish-

able iff there exists some input segment $\underline{x} \in X$ such that $\bar{T}(\alpha_1 ; \underline{x})$ is not identical to $\bar{T}(\alpha_2 ; \underline{x})$. In symbols,

$$s_1 \neq s_2 \Leftrightarrow \underline{x} \in X \text{ s.t. } \bar{T}(\alpha_1 , \underline{x}) \neq \bar{T}(\alpha_2 , \underline{x}). \tag{22}$$

Thus two distinct states may be either equivalent or distinguishable. The negation of this statement is, of course, not true. In other words, two identical states $s_1 = s_2$ must be equivalent states. Hence, equivalence does not imply identity, nor does distinctness imply non-equivalance (i.e., distinguishability). Interest in state equivalence is motivated by the desire to construct *reduced objects*, or objects in the *reduced form*, which are defined as objects no two states of which are equivalent (i.e., every pair of states are distinguishable). We then speak of the state space S as being "*minimal*" and denote it by $S_{\min}$.

More fundamentally, we are really interested in defining α-states to the highest possible degree of aggregation while still satisfying the three consistency conditions (Conditions I to III) stated above. The greatest advantage of such aggregation is, obviously, to minimize the *number* of α-states that must be enumerated and investigated. As is well known in the applications of DP models, the computational burden increases multiplicatively with the increase in the number of distinguishable elements of S.

For example, in the Traveling Salesman Problem, it is possible to define two α-state spaces, say S and S', as follows:

(i) In S, an α-state α_m is a subset of cities containing city $\underline{1}$, m cities from the set $B = \{2, 3,..., n\}$ *appearing in the order of their sequence*, such that the last city visited is denoted by i_m. That is, α_m is a vector of $m + 1$ entries

$$\alpha_m \triangleq \{(1, i_1 i_2 \cdots i_m)\}, \qquad i_j \neq i_k , \; 1 \leqslant j, k \leqslant m \leqslant n - 1.$$

(ii) In S', an α-state α_m is simply a subset of $m + 1$ cities always contains city $\underline{1}$; $m \leqslant n - 1$, and the last city visited, i_m.

Clearly, the space S contains many more α-states than S'. It can be shown that with the proper definition of the value function $v(\pi)$, both spaces satisfy all three consistency conditions, but S contains many equivalent states while S' is really a minimal state space.

The Construction of Classes of Equivalent States

It is evident from the above discussion that if one is given the object in terms of the input-output relation $\mathcal{O}(x, v) = 0$, then one attempts to construct the minimal state space, $S_{\min}$, which satisfies all three consistency conditions.

On the other hand, if one is *given* a state space S, together with the input-output-state equation (i.e., $\mathcal{O}$ is defined as in Eq. (7) or (8)), then there is

always the question of reducing S to $S_{\min}$. This necessitates the determination of the set of equivalent states to every state $s \in S$.

The "long hand" way of accomplishing this objective is to: (i) choose $s_0 \in S$; (ii) apply an input segment $\underline{x} \in X$ and determine the state segment $\bar{T}(\alpha_0, \underline{x})$; (iii) apply the *same* input segment $\underline{x}$ to every other state $s' \in S$ and determine the corresponding state segment $\bar{T}(\alpha', \underline{x})$; (iv) all the states $s' \in S$ whose $\bar{T}(\alpha', \underline{x}) \equiv \bar{T}(\alpha_0, \underline{x})$ are equivalent to s_0; and we denote the class of equivalent states by $[s_0]$, where s_0 is any member of the class. This "long hand" way is really a verbalization of the definition (22).

In this light, we shall always refer to the "state of the object $\mathcal{O}$ at stage n" to mean *any* state in the class of equivalent states given by $Q_n(s_0, x)$ of Eq. (13).

THEOREM 2 (*The State Separation Property* (*State-SP*)). *For any input consisting of a segment* $\underline{x}_1 = a_1 a_2 \cdots a_m$ *followed by a feasible continuation segment* $\underline{x}_2 = a_{m+1} a_{m+2} \cdots a_n$, *the state segment of* $\mathcal{O}$ *corresponding to the input* $\underline{x}_1 \underline{x}_2$ *starting in state* s_0 *consists of the state segment* $\bar{W}(s_0, \underline{x}_1)$ *followed by the state segment* $W(s_m, \underline{x}_2)$, *where* s_m *is the state of* $\mathcal{O}$ *at the end of stage* m.

Proof. From Condition III, and in particular Eq. (12),

$$s_m \in Q(s_0, x_1 x_2),$$

where $\underline{x}_1 = a_1 a_2 \cdots a_m$, $\underline{x}_2 = a_{m+1}, a_{m+2}, \ldots, a_n$.
By the same Condition, if we let $\underline{x}_1$ be as shown above, $\underline{y}_2 = a_{m+1} a_{m+2} \cdots a_r$ and $\underline{y}_3 = a_{r+1} a_{r+2} \cdots a_n$, we have

$$s_r \in Q(s_0 ; x_1 y_2), \qquad n > r > m.$$

But appealing to Condition II, we have

$$v_m = G(s_0 ; x_1), \; v_r = G(s_0 ; x_1 y_2) \text{ and } v_n = G(s_0 ; x_1 x_2) = G(s_0 ; x_1 y_2 y_3).$$

Furthermore, the state-equation (15) results in

$$s_m = W(s_0 ; x_1), \qquad s_r = W(s_0 ; x_1 y_2) \qquad \text{and} \qquad s_n = W(s_0 ; x_1 x_2),$$

from which we deduce that

$$\underline{v}_1 = \bar{G}(s_0 ; \underline{x}_1), \; \underline{v}_2 = \bar{G}(s_m ; \underline{y}_2), \; \underline{v}_3 = \bar{G}(s_r ; \underline{y}_3), \text{ and } \underline{v} = \underline{v}_1 \circ \underline{v}_2 \circ \underline{v}_3 .$$

Now consider s_r as a function of s_m and y_2; it must be true that

$$W(s_m ; y_2) = s_r , \tag{23}$$

for, if not true, and $W(s_m ; y_2) = s_r' \neq s_r$, then the output segment between stages r and n would be $\underline{v}_3' = \bar{G}(s_r ; \underline{y}_3) \neq \underline{v}_2$ as defined above, and

$y_3 = a_{r+1}a_{r+2} \cdots a_n$, which contradicts the uniqueness of the output segment $\underline{v}$ and Eq. (12). This establishes the validity of (23), and we have

$$s_r = W(s_m ; y_2) = W(s_0 ; x_1 y_2).$$

Iterating over all r, $m + 1 \leqslant r \leqslant n$, we conclude that

$$\underline{s}_n = \overline{W}(s_0 ; \underline{x}_1) \circ \overline{W}(s_m ; \underline{x}_2);$$

where the right hand expression indicates the concatenation of state segments. Theorems 1 and 2 establish a fundamental property of SDDP's; viz., that the state s_n at any stage n (i.e., at the end of the application of the input segment $\underline{x} = a_1 a_2 \cdots a_n$ to $\mathcal{A}$ at initial state s_0) as well as the value of the output, v_n, at that stage can be evaluated as functions W and G of any state s_i and the input segment $a_{i+1}a_{i+2} \cdots a_n$ for all $1 \leqslant i \leqslant n$, respectively. Needless to say, the SP for state and output is the cornerstone of the analysis of SDDP's.

The Concept of Equivalent Policies

Theorem 1 and subsequent development led to the definition of equivalent *states*, see Def. (21). It is meaningful to inquire about the concept of equivalent *policies*.

Notice that by direct appeal to the input-output-state equation, Eq. (7), the value of the output at stage m, which we denoted by v_m, is a function of s_0 and $\underline{x}$ *only*, i.e.,

$$v_m = G(s_0 ; x); \qquad \underline{x} = a_1 a_2 \cdots a_m$$

Suppose another input $x' = a_1' a_2' \cdots a_n'$ is applied to $\mathcal{A}$ at the *same* initial state s_0 ; we could then obtain the output

$$v_n' = G(s_0 ; x').$$

Clearly, v_n' need not bear any relationship to v_m. Let $s_n' = W(s_0 ; x')$. Then if $v_n' = v_m$, we say that policies x and x' yield the same terminal output. (Notice that the output segment $\underline{v}'$ may not be equal to the output segment $\underline{v}$ at all intermediate stages.) If *in addition* $s_n' \simeq s_m$, then policies x' and x terminate at the *same* state, i.e., s_m must be identical to s_n ; $s_m \equiv s_n$. In this case we are led to:

DEFINITION (*Equivalent Policies*). Two distinct policies $\underline{x} = a_1 a_2 \cdots a_m$ and $\underline{x}' = a_1' a_2' \cdots a_n'$ are said to be *equivalent relative to the initial state* s_0, written $x \simeq_{s_0} x'$, iff (i) $v_m \triangleq G(s_0 , x) = G(s_0 , x') \triangleq v_n'$ and (ii) $W(s_0 , x) \simeq W(s_0 , x')$.

Notice that equivalence between *policies* is *dependent* on the initial state of $\mathcal{A}$ at which the policies are applied (i.e., x_1 and x_2 may be equivalent if $\mathcal{A}$ is

initially in state s_0 but are not equivalent if $\mathcal{A}$ is initially in state $s_0' \neq s_0$), while the equivalence between *states* was defined over all continuation policies (and hence was *independent* of continuation policies applied to $\mathcal{A}$).

Furthermore, equivalence of *policies* is defined in terms of the *terminal state and terminal output*, not in terms of segments of states and segments of outputs. This is in stark contrast to all previous definitions and relations.

COROLLARY 1. *If two states s_m and s_n, which are arrived at from initial state s_0 by two distinct policies $\underline{x}_1 = a_1 a_2 \cdots a_m$ and $\underline{x}_2 = a_1' a_2' \cdots a_n'$, respectively, are* equivalent, *then it is always possible to make one of the following two statements concerning the final value of the output v_r under all permissible continuations*

$$\underline{y} = \underline{a}_1'' a_2'' \cdots a_r'' \in X_{s_m} \cap X_{s_n} \qquad \text{of } x_1 \text{ and } x_2\text{:}$$

either

$$v(s_0 ; x_1 y) \geqslant v(s_0 ; x_2 y) \tag{24}$$

or

$$v(s_0 ; x_1 y) \leqslant v(s_0 ; x_2 y).$$

Proof. Apply first the output-SP property, then apply theorem 1 followed by the definition of state equivalence to obtain

$$(v_r \,|(s_0 , x_1 , y)) = G[(v_m ; \alpha_m); y]$$

and

$$(v_r \,|(s_0 , x_2 , y)) = G[(v_n ; \alpha_m); y].$$

Obviously, since G is a mapping into V (which is a linearly ordered metric space), the two values are equal if, and only if, $v_m = v_n$, i.e., if $x_1 \simeq_{s_0} x_2$. Otherwise one of the two values must be greater than the other.

Remark 4. Interestingly enough, the result in (24) is a fundamental element in the necessary and sufficient conditions for a DDP to be represented as an SDDP given by Karp and Held (see their Theorem 2, Condition (ii)). In our development, it appears as a corollary to the definition of the state space. In other words, if a state space is defined as given above, this "condition" of Karp and Held is *always* satisfied.

COROLLARY 2. *Consider two equivalent states: $s_1 = (v_1 , \alpha_1)$ and $s_2 = (v_2 , \alpha_2)$, $s_1 \simeq s_2$. If $v_1 = v_2$ then $\bar{G}(s_1 ; \underline{x}) \equiv \bar{G}(s_2 ; x)\ \forall\, x \in X$.*

Another way to look at these two corollaries is to consider a state s which is reached from s_0 by two distinct policies $x_1 \triangleq a_1 a_2 \cdots a_m$ and $x_2 \triangleq a_1' a_2' \cdots a_n'$; i.e.,

$$s = W(s_0 ; x_1) = W(s_0 , x_2) \equiv (v; \alpha).$$

Let

$$v_m^{(1)} \triangleq G(s_0 ; x_1)$$
$$v_n^{(2)} \triangleq G(s_0 ; x_2).$$

Assume that $v_m^{(1)} = v_n^{(2)}$. Under a feasible (elementary decision) continuation of x_1 and x_2, we have that

$$T(\alpha_0 ; x_1 a) = T(\alpha_0 ; x_2 a) \equiv T(\alpha; a).$$

Let $s_1 = W(s; a) \equiv (v_1 , \alpha_1)$; then clearly

$$v_1 = G(s_0 ; x_1 a) = G(s; a) \equiv G(s_0 ; x_2 a)$$

and $\alpha_1 = T(\alpha; a)$.

Iterating over any feasible continuation policy we conclude that the value of the output from state s onwards will be identical (see Fig. 2).

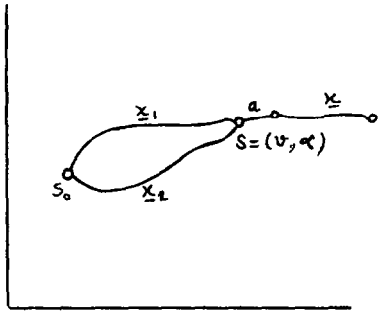

FIGURE 2

FURTHER PROPERTIES OF THE STATE SPACE

THEOREM 3 (*The Sufficiency of the Output-SP to Define S*). *If the output* v_m *defined by the input-output-state equation*

$$v_m = G(s_0 , x); \; s_0 \in S; \; x \in X; \; \underline{x} \triangleq a_1 a_2 \cdots a_m \tag{25}$$

in which $G : S \times X \to V$ *and* G *has the output-SP, i.e., the output segment*

$$\underline{v} = \bar{G}(s_0 , \underline{x}) = \bar{G}(s_i , \underline{x}''), \quad \forall \, \underline{x}'' = a_{i+1} a_{i+2} \cdots a_m , \quad i = 1, 2, \ldots, m - 1, \tag{26}$$

and s_i *dependent only on* s_0 *and* $x' = a_1 a_2 \cdots a_i$,
then (26) *satisfies Consistency Conditions* II *and* III *for some object* $\mathcal{O}$, *and* s_i *qualifies as the* state *of* $\mathcal{O}$ *at stage* i, *with* s_0 *being the initial state of* $\mathcal{O}$. *If* (26)

is true for all initial states $s_0 \in S$, then S is a state space of $\mathcal{A}$. (Note that since $\mathcal{A}$ was not defined by an input-output relation, there is no need to consider the Mutual Consistency Condition, Condition I.)

Proof. Condition II is obviously satisfied by the fact that the output v is uniquely determined by s_0 and x in (25). As to Condition III, the separation property (26) specifies that s_i depends only on s_0 and $\underline{x}' \triangleq a_1 a_2 \cdots a_i$. Thus every set $Q(s_0 ; x'x'')$ will contain s_i, and hence their intersection (Eq. (13)) is not empty by virtue of containing (at least) s_i. This proves the theorem.

THEOREM 4 (*The Sufficiency of the State-SP to Define S*). *If the state-segment $\underline{s}_n$, defined by the state-equation* (16) *has the state-SP, i.e., if the state segment can be expressed as*

$$s_n = \overline{W}(s_0 ; \underline{x}_1\underline{x}_2) = \overline{W}(s_0 ; \underline{x}_1) \circ \overline{W}(s_m ; \underline{x}_2); \forall\, s_0 \in S;\ \underline{x}_1\underline{x}_2 \in X; \tag{27}$$

$$\underline{x}_1 \triangleq a_1 a_2 \cdots a_m ;$$

$$x_2 \triangleq a_{m+1} a_{m+2} \cdots a_n ; \tag{28}$$

$$s_m = W(s_0 ; x_1) \in S;$$

and if the state s_n is always a vector of the form

$$s_n = (v_n ; \alpha_n) \tag{29}$$

where

$$v_n = G(s_0 ; x_1 x_2); \tag{30}$$

then the object $\mathcal{A}$ defined by the spaces X, V and S satisfies Condition II *to* IV. *(Note that since $\mathcal{A}$ was not defined by an input-output relation, there is no need to consider the Mutual Consistency Condition, Condition* I.)

Proof. Condition II is satisfied because v is a (point) function of s_0 and x, from Eq. (30). Condition III is satisfied by the fact that s_m is dependent only on s_0 and $\underline{x}_1$ and is independent of all extensions $\underline{x}_2$, see Eq. (28); i.e., the set $Q(s_0 ; \underline{x}_1)$ is not empty. Condition IV is satisfied by hypothesis, see Eqs. (27) and (29).

Remark 5. The difference between Theorems 3 and 4 lies in the fact that the output-SP is sufficient to satisfy Conditions II and III but *not Condition* IV. However, a study of Theorems 1 and 2 and the intervening as well as the subsequent discussion reveals that Condition IV is not needed for the *definition* of the state space S, but only in deriving certain properties of that space (such as the concept of equivalent states and the transformation function T). Consequently, Condition IV must be added to the output-SP to specify

completely all the desired properties of S. This was done in Theorem 4, in which Condition IV was included in the hypothesis of the theorem.

The upshot of both theorems is that the output-SP plus the Decomposition Property of the state, as well as the state-SP plus the same Decomposition Property, are each necessary and sufficient conditions for the complete definition of $\mathcal{O}$ as a SDDP.

In other words, combining Theorems 1 and 3 we conclude that the output-SP is necessary and sufficient for the definition of the state space S when the object $\mathcal{O}$ is defined in terms of its input-output-state equation. If $\mathcal{O}$ is also defined in terms of its input-output relations, then the output-SP *and* Condition I are necessary and sufficient for defining S.

The same statement can be repeated verbatim relative to the state-SP.

The importance of Theorems 3 and 4 stems from the fact that they reduce the burden of verifying the self-consistency conditions to that of demonstrating that the input-output-state equation (25) has the output-SP or the state-SP.

Remark 6. Let the output v of an object $\mathcal{O}$ be characterized by the input-output-state equation

$$v = G(s, x), \qquad s \in S, x \in X.$$

Let M be a one-one mapping of the state space S onto itself, $M : S \to S$; which takes s into s'; i.e.,

$$M(s) = s' \qquad \text{and} \qquad M^{-1}(s') = s,$$

where M^{-1} is the inverse mapping of M. Then the equation

$$v = G(s', x) \triangleq G(M(s); x)$$

is also an input-output-state equation of $\mathcal{O}$.

The assertion must be true because the mapping M is merely a relabeling of the states and did not alter their definition, their space, or any of their properties.

Properties of "Descendants" of s

These properties will be presented as a number of theorems.

THEOREM 5. *Let S_s denote the set of all descendants of s.* Then *S_s is a* closed set *in the sense that if s_0 is any initial state in S_s, then every state which is reachable from s_0 is also in S_s.*

Proof. If s' is reachable from $s_0 \in S_s$, then, by definition, there exists an input segment $\underline{x}'$ such that $s' = W(s_0, x')$. Since $s_0 \in S_s$ then, by definition of S_s, there exists an input segment $\underline{x}$ such that $s_0 = W(s, x)$. Consider now

the composite input $\underline{x}\underline{x}'$ applied to $\mathcal{A}$ in initial state s. Clearly, $W(s, xx') = W(s_0, x')$, by the state-SP. But, by assumption, $W(s_0, x') = s'$; hence s' is reachable from s; i.e., $s' \in S_s$. Since this is true for all s, s_0, s', the theorem is proven.

THEOREM 6. If $s_1 \simeq s_2$ and $s' = (v_m', \alpha')$ and $s'' = (v_m'', \alpha'')$ are two *states reached from* s_1 *and* s_2 *by the* same input segment x; *respectively, then* $s' \simeq s''$. *In other words, states reached from equivalent states by the same input are themselves equivalent.*

Proof. By contradiction. Suppose that $s' \not\simeq s''$. Then there exists an input segment $\underline{x}'$ which distinguishes between s' and s''. Now consider the composite input $\underline{x}\underline{x}'$ applied respectively to s_1 and s_2. By the state-SP, we have that

$$\overline{W}(s_1 ; \underline{x}\underline{x}') = \overline{W}(s_1 ; \underline{x}) \circ \overline{W}(s', \underline{x}')$$

and

$$\overline{W}(s_2 ; \underline{x}\underline{x}') = \overline{W}(s_2 , \underline{x}) \circ \overline{W}(s'', x'). \tag{31}$$

Since $s_1 \simeq s_2$, we conclude that $\overline{T}(\alpha_1 , \underline{x}\underline{x}') \equiv \overline{T}(\alpha_2 , \underline{x}\underline{x}')$ by the definition of equivalent states, see (21). And, by the same argument, $\overline{T}(\alpha_1 ; \underline{x}) \equiv \overline{T}(\alpha_2 ; \underline{x})$. Substituting these two equalities in (31), we conclude that

$$\overline{T}(\alpha', \underline{x}') \equiv \overline{T}(\alpha'', \underline{x}');$$

which contradicts the assumption of distinguishability of α' and α'' under $\underline{x}'$. This proves the theorem.

Remark 7. If the SDDP is in *reduced form*, no two states s_1 and s_2 are equivalent. Hence, an input segment $\underline{x}$ applied to any two states will yield two nonequivalent states. Consequently, in reduced objects a stronger statement can be made; to wit, if $s' \equiv (v_{m'}, \alpha')$ and $s'' \equiv (v_{m''}, \alpha'')$ are two states reachable from $s_1 \equiv (v_1 , \alpha_1)$ and $s_2 \equiv (v_2 , \alpha_2)$ by the same input segment $\underline{x}$, then $s' \simeq s''$ if, and only if, $s_1 \simeq s_2$.

The proof of necessity is similar to the proof of Theorem 6. The proof of sufficiency is obtained by remarking that T is a measurable mapping, T: $S \times X \to S$, hence its inverse T^{-1}, is also a measurable mapping. Consequently, by the fact that $\mathcal{A}$ is in reduced form,

$$s' \simeq s'' \Rightarrow s' \equiv s'';$$

Consequently,

$$T^{-1}(s', x) \equiv T^{-1}(s'', x);$$

from which we conclude that $s_1 \equiv s_2$, which implies that $s_1 \simeq s_2$.

Remark 8. Notice that the theorem does *not* exclude the possibility of an input x to an object $\mathcal{O}$ starting at two distinguishable states $s_1 \not\simeq s_2$ resulting in states $s' \simeq s''$, i.e.,

$$s_1 \not\simeq s_2 \qquad \text{but } \alpha'' \triangleq T(\alpha_1, x) \equiv T(\alpha_2, x) \triangleq \alpha''.$$

Furthermore, the theorem does not make any implication about two *different inputs* x_1 and x_2 applied to $\mathcal{O}$ at two distinguishable states s_1 and s_2, respectively, whether these states are equivalent or not.

Monotone Terminating Processes (or Finite Monotone Objects)

Our interest here is in a special class of SDDP's, namely, those which "terminate" in a finite number of stages, and which have the property that they are "monotone" processes. We proceed to make these notions precise.

Definition (*Final States of* $\mathcal{O}$). A state $t \in S$ is said to be a *final state* iff

$$T(\alpha_t, a) = \phi \; \forall \; a \in \mathcal{A}. \tag{32}$$

The set of all final states satisfying (32) is the set $F \subset S$.

It is an immediate conclusion that if $t \in F \subset S$ then $T(\alpha_t, x) = \phi$ for all input policies $x \in X$.

Definition (*Monotone Objects*). An object $\mathcal{O}$ is said to be a *monotone* object, or a monotone sequential decision process, MSDP, if the following condition is satisfied: consider $\mathcal{O}$ initially in state $s_0 \equiv (v_0, \alpha_0)$, and let policies x_1 and x_2 be applied, changing the state of $\mathcal{O}$ to $s_1 \equiv (v_1, \alpha_1)$ and $s_2 \equiv (v_2, \alpha_2)$, respectively. Let the policy x' be a feasible continuation of either x_1 or x_2, with $x' \equiv a_1' a_2' \cdots a_n'$. If $v_1 \geqslant v_2$, then

$$G((v_1, \alpha_1); a_1') \geqslant G((v_2, \alpha_2); a_1')$$

and, by induction on $L(\underline{x}')$,

$$G((v_1, \alpha_1), x') \geqslant G((v_2, \alpha_2); x') \; \forall \; x' \in X.$$

As we shall discover presently, we are obliged to assume that our SDDP's are monotone processes, in order to be able to apply the powerful theory of Dynamic Programming. The monotonicity assumption was first introduced by Mitten [9] as a *sufficient* condition for the well-known "principle of optimality of DP." Prior to Mitten's work, the subtleties of that "principle" were largely unexplored.

Definition (*Terminating SDDP's*). We define a *terminating SDDP* as an object possessing the following two properties:

(i) The set F of final states is non-empty.

(ii) For any state $s \in S$ and any permissible policy $x \in X_s$ of finite length, either $W(s, x) \in F$ or there exists a permissible continuation policy $x' \in X$ of finite length such that $W(s, xx') \in F$.

In other words, terminating SDDP's must "stop" after a finite number of stages, no matter what the starting state of $\mathcal{O}$ might have been.

We now come to the central result of this investigation, viz., the characterization of the necessary and sufficient conditions for representation of a DDP by an SDDP. The importance of the result derives from the close connection between MSDP's and the dynamic programming approach.

THEOREM 7. *A monotone sequential decision process exists iff the state space S has the state-*SP *and the object $\mathcal{O}$ is monotone.*

Proof. An SDDP, monotone or not, satisfies Conditions I to III by construction; which implies, by Theorem 2, that it possesses the state-SP. This proves the necessity of the condition. The proof of sufficiency follows from the fact that a sequential decision process with the state-SP satisfies Conditions I to III, by Theorem 4, and hence is an SDDP; and if it also satisfies the monotonicity conditions, then we have an MSDP.

RELATION TO DYNAMIC PROGRAMMING

Except for a fleeting reference in the Introduction, we have confined our discussion to the *analysis* of a given, or derived, structure and the delineation of its properties. The notion of *optimization* has been studiously kept in the background.

However, it is high time to return to the notion of optimization, because that is what dynamic programming (DP) is all about. In particular, DP is concerned with the optimization of sequential decision processes. That is, DP is the theory of determining the maximum (or minimum) or the output of a sequential decision process, when the variables of the process are constrained either individually or in the aggregate.

We limit our attention to *discrete*, finite-state sequential decision processes with finite $X_s \forall s \in S$, and in particular to terminating MSDP's, as defined above possessing the state decomposition property. So, for our immediate purposes, we take DP to be an approach to the optimization of constrained but terminating MSDP's. Because of the assumptions of finite policies and finite stage to termination, there is no question of the existence of a maximum (or a minimum) output. As we shall presently see, DP capitalizes on the properties of the state which were detailed above. Indeed, this is the main

reason for deriving the structure of MSDP's from DDP's in the first place. As was mentioned before, the majority of decision problems occur "au naturel" as DDP's, while DP is relevant only to SDDP's—hence the need to bridge the gap from one formulation to the other.

Let the starting state be s_0. Consider the problem of maximizing the value of the output, v, over all feasible input policies $x \in X_{s_0}$ (the Fixed-Initial-State Problem, also sometimes referred to as the Initial-Value Problem). For any fixed n; $1 \leqslant n \leqslant M < +\infty$; let $x_n \triangleq a_1 a_2 \cdots a_n \in X_{s_0}$. Let $F_n(s_0)$ denote the maximum value of the output over all permissible policies x_n of length $L(x_n) = n$. Then we have,

$$\begin{aligned} F_n(s_0) &\triangleq \max_{x_n} G(s_0 ; x_n); \qquad x_n \in X_{s_0} \\ &= \max_{x_n} G(s_{n-1} ; a_n) \text{by the value-SP} \\ &= \max_{x_n} G(v_{n-1} , \alpha_{n-1} ; a_n) \text{ by the state-decomposition property} \\ &= \max_{x_n} G[G(s_0 ; x_{n-1}), \alpha_{n-1} ; a_n]; \qquad x_{n-1} = a_1 a_2 \cdots a_{n-1} . \end{aligned} \tag{33}$$

But, by the monotonicity assumption, if policy x_{n-1} yields the pair v_{n-1}, α_{n-1}, while policy x'_{n-1} yields the pair v'_{n-1}, α_{n-1}, such that $v_{n-1} \geqslant v'_{n-1}$, then $G(v_{n-1} , \alpha_{n-1} , a_n) \geqslant G(v'_{n-1} , \alpha_{n-1} , a_n)$. In particular,

$$G(v^*_{n-1} , \alpha_{n-1} , a_n) \geqslant G(v_{n-1} , \alpha_{n-1} , a_n), \forall v_{n-1} = G(s_0 , x_{n-1}),$$

where $v^*_{n-1} = \max_{x_{n-1}} G(s_0 ; x_{n-1}) \triangleq F_{n-1}(s_0); x_{n-1} \in X_{s_0}$.

Substituting above in (33), we obtain

$$F_n(s_0) = \max_{a_n} G[F_{n-1}(s_0); \alpha_{n-1} ; a_n] \tag{34}$$

which is a general form of the well-known functional equation of DP.

Since, by assumption, the sequential process is terminating in a finite number of stages, no matter where its initial state happens to be, one can always employ a (backward) recursive procedure to determining $F_n(s_0)$, which is the maximum output when the process starts in state s_0.

If the starting state is not given (the so-called Variable Initial-State Problem), but any state in a subset of states $I \subset S$ is eligible as a starting state, then the values $(F_n(s_0))$ are evaluated for every $s_0 \in I$ and the optimum over these values is determined.

Application of DP to Feedback Processes

The notion of state, and the subtle and important role it plays in the formulation of SDP's, is best illustrated by the attempted application by Rudd

and Blum [11] of the "functional equation of DP" — which implies the "Principle of Optimality" — to a chemical process characterized by a feedback loop.

The process was that of extracting a solute from a solvent by addition of wash water at each of $n = 3$ stages, as shown in Fig. 3. The solvent and wash

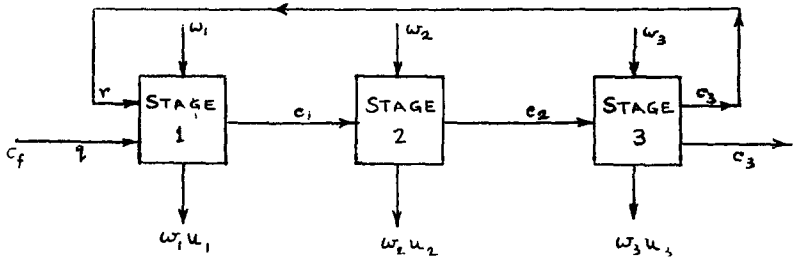

FIG. 3. 3-Stage cross current extraction.

water are assumed immiscible. The solvent flows from stage to stage at a rate $q + r$. The concentration of the solute as it leaves stage i is c_i, while the concentration of the solute in the wash water leaving stage i is u_i. All concentrations (of input, output, and water overflow) are in equilibrium,

$$\begin{aligned} (q + r)\, c_{i-1} &= (q + r)\, c_i + w_i u_i \\ u_i &= \psi(c_i) \qquad\qquad i = 1, 2, 3 \\ c_0 &= (qc_f + rc_N)/(q + r) \end{aligned} \tag{35}$$

where c_f is the concentration in the feed solvent, and w_i is the quantity of water flowing, in a cross-current pattern, at stage i, and ψ is some (nonlinear) function relating u_i to the input concentration c_i. A portion of the end product is fed back to the first stage at a flow rate r. The decision variable at each stage i is the amount of wash water w_i, used at that stage. The objective is to

$$\text{maximize } P = q(c_f - c_N) - \lambda \sum_i w_i$$

subject to the equilibrium equations stated above. Here, λ represents the cost of providing a unit of wash water.[4]

The approach by Rudd and Blum can be summarized as follows. Let $\{w_i^*(c_0)\}$ represent the set of input water flows which maximize P for a given c_0. These values may be found by conventional DP computations in a

[4] The same extraction problem *without* recycle was successfully treated by Aris, Rudd, and Amundson [1], in which c_i was the state of the system.

"forward" movement from stage 1 to stage 3, and in turn they determine a value of c_3,

$$c_3 = c_3[\{w_i^*(c_0)\}, c_0].$$

The pair of equations relating c_0 and c_3 may then be solved simultaneously for c_0 and c_3, which suffice to determine the optimum (w_i^*) and the maximum P. Rudd and Blum have also suggested an iterative procedure for determining the optimum values of the decision variables.

The fallacy of this approach was demonstrated by Jackson [6] in a Letter to the Editor. His arguments were based on the characteristics of a stationary point which is the maximum. He provided a simple counterexample which would yield the *least* desirable result if attacked by the DP approach.

The same cross-current extraction problem was solved (optimally) by Fan and Wang [5], who used arguments based on Pontryagin's Maximum Principle.

Thus, there is no doubt that the Rudd and Blum DP approach to this problem is invalid. There is also no doubt that the correct solution can be obtained through application of the Maximum Principle. Our interest really lies in: why did "straightforward" DP fail? This necessitates a closer look at the problem in light of our theory.

The basic error in "fixing" c_0, the input "state" to stage 1, and carrying out the partial optimization of the stages in traditional DP fashion as a function of c_0 is that such an approach assumes that the terminal state of the process is freely chosen from a subset of feasible states. This is the standard approach to the "initial value problem" of DP. In fact, this is not the case since the terminal state of our process, denoted by c_3, is constrained to bear a very specific relationship to the initial state c_0. In particular, for any "fixed" c_0 the terminal state is given *uniquely* by

$$c_3 = [c_0(q + r) - qc_f]/r.$$

It is well known, and, anyway, can be easily demonstrated (see, for example, ref. [17], p. 385), that it is incorrect to carry out optimization without regard to constraints except at the very end.

Having said that, we now show that c_0 cannot function as the "state" of the system at stage 1 in the sense of our definitions of state (see Eq. (7)). To this end, it is sufficient to show that c_0 cannot parametrize the input-output relations in the manner indicated before to permit the translation of the decision problem to a *sequential* decision problem amenable to treatment by DP.

Suppose we divide the concentration balance equation (the first equation of Eq. (35)) by $(q + r)$, then we can reformulate the problem as:

$$\text{maximize } P = q(c_f - c_3) - \lambda(q + r)\sum_i v_i, \qquad v_i = w_i/(q + r)$$

such that

$$\begin{aligned} c_i &= c_{i-1} - v_i u_i \\ u_i &= \psi(c_i) \qquad\qquad i = 1, 2, 3 \\ c_0 &= \alpha c_f + (1 - \alpha)\, c_3 ; \qquad \alpha = q/(q + r). \end{aligned} \tag{36}$$

Apparently, this is a decision process with four decision variables (the four variables which appear in the objective function); the output is, of course, P. However, a little reflection reveals that only *three* of these four variables can be arbitrarily determined; the fourth is then uniquely determined from the material balance constraints. In particular, if all (v_i) are specified, there is a (not necessarily unique) solution of the remaining 7 equations in 7 unknowns $(c_0, c_1, c_2, c_3, u_1, u_2, u_3)$. The nonlinearity of these equations may render their solution a difficult task indeed[5] but one can achieve such a solution, at least in principle. The three variables v_1, v_2, and v_3 are precisely what was referred to previously as the "input," $\underline{x}$. Consequently, the representation of the decision process by a sequential decision process necessitates the parametrization of the input-output relation, i.e., directed object, $\mathcal{O}(P, v_1, v_2, v_3) = 0$. But, obviously, such parametrization is not possible in terms of c_0 because c_0 is itself uniquely determined by the "input" vector (v_1, v_2, v_3). Or, put differently, if one specifies (i.e., hypothesizes) a value of c_0 one automatically eliminates choice in one of the three decision variables. This, in turn, seriously detracts from the *optimization* process.

The question that arises is: can this decision problem be interpreted as a sequential decision problem at all? The answer, fortunately, is: yes.

Of course, at this juncture, the analyst is free to attack the decision problem directly, which involves search in the three-dimensional space of v_1, v_2 and v_3. Or he may elect to apply DP to the problem after defining the "state" correctly. As it turns out, in a three-stage feedback loop as we have in our example, it seems that the computational effort is approximately the same for both approaches.

The DP approach follows the concepts of "cut-state" and "decision inversion" advanced by Wilde [12]. Consider a fixed value of c_3 (the "cut-state"), which will be denoted by $\tilde{c}_3$. The "decision inversion" involves the fixing of the value of c_2, then, since $\tilde{u}_3 = \psi(\tilde{c}_3)$, we have $\tilde{v}_3 = (c_2 - \tilde{c}_3)/\tilde{u}_3$, a function of c_2. Since Stage 3 is now a decisionless stage, it can be lumped with Stage 2. First we note that

$$c_2 = c_1 - v_2 u_2 = c_1 - v_2 \cdot \psi(c_2),$$

[5] Notice that even if the function ψ were linear, the end result would still be a rational fraction of two polynomials in the v_i's.

from which we can, at least in principle, invert the relationship and obtain an expression of c_2 in terms of c_1 and v_2,

$$c_2 = h_2(c_1, v_2).$$

Consequently, the last two stages return function is given by

$$\begin{aligned} P_2 &= q(c_f - \tilde{c}_3) - \lambda(q + r)(v_2 + \tilde{v}_3) \\ &= q(c_f - \tilde{c}_3) - \lambda(q + r)[v_2 + (h_2(c_1, v_2) - \tilde{c}_3)/u_3], \end{aligned}$$

a function of $\tilde{c}_3$, c_1, and v_2. Optimizing P_2 over the decision variable v_2 yields a function of $\tilde{c}_3$ and c_1 only; i.e.,

$$\begin{aligned} f_2(c_1, \tilde{c}_3) &= \max P_2(c_1, \tilde{c}_3, v_2). \\ &= P_2^*(c_1, \tilde{c}_3, v_2^*), \end{aligned}$$

where $v_2^* = v_2^*(c_1, \tilde{c}_3)$, a function of c_1 and $\tilde{c}_3$.

Finally, we arrive at stage 1, which must be handled carefully because its *input state* c_0 is predetermined by the assumed value of $\tilde{c}_3$. In particular, we have a one-decision, no-state optimization problem:

$$f_1(\tilde{c}_3) = \max_{v_1}\{-\lambda(q + r)\, v_1 + f_2(\tilde{c}_3 ; c_1)\}.$$

Remarking that

$$c_1 = c_0 - v_1 u_1 = \alpha c_f + (1 - \alpha)\, \tilde{c}_3 - v_1 \cdot \psi(c_1),$$

we can, at least in principle, invert the relationship to obtain c_1 as a function of v_1 and $\tilde{c}_3$; i.e.,

$$f_1(\tilde{c}_3) = \max_{v_1}\{-\lambda(q + r)\, v_1 + f_2(\tilde{c}_3 ; h_1(\tilde{c}_3, v_1))\}.$$

Finally, the optimal of the feedback system is obtained from

$$f_1 = \max_{\tilde{c}_3} f_1(\tilde{c}_3).$$

In this correct DP formulation, *the state at any intermediate stage i is given by* $(c_i, \tilde{c}_N)$ *and not by* c_i *alone* as in the previous attempt.

References

1. R. Aris, D. F. Rudd, and N. R. Amundson. The optimum cross-current extraction. *Chem. Eng. Sci.* **12** (1960), 88–97.
2. R. Bellman. "Dynamic Programming." Princeton Univ. Press, Princeton, N.J., 1957.

3. B. W. BROWN. On the Iterative Method of Dynamic Programming on a Finite Space Discrete Time Markov Process. *Ann. Math. Stat.* **26** (1965).
4. E. V. DENARDO AND L. G. MITTEN. Elements of sequential decision processes. *J. Ind. Eng.* **18** (1967).
5. L. T. FAN AND C. S. WANG. "The Discrete Maximum Principle." p. 37. Wiley, New York, 1964.
6. R. JACKSON. Comments on the paper: Optimum cross-current extraction with product recycle. Letter to the Editor. *Chem. Eng. Sci.* **18** (1963), 215–217.
7. R. M. KARP AND M. HELD. Finite state processes and dynamic Programming. *J. SIAM* **15** (1967).
8. M. D. MESAROVIC, ED. "Views on General Systems Theory." Wiley, New York, 1964.
9. L. G. MITTEN. Composition principles for synthesis of optimum multi-stage processes. *Oper. Res.* **12**, No. 4, July–August, 1964.
10. G. L. NEMHAUSER. Introduction to dynamic programming." Wiley, New York, 1966.
11. D. F. RUDD AND E. D. BLUM. Optimum cross-current extraction with product recycle. *Chem. Eng. Sci.* **17** (1962), 277–281.
12. D. J. WILDE. Optimizing macro-systems. *Chem. Eng. Progr.* **61** (March 1965), 86–93.
13. D. J. WILDE AND C. S. BEIGHTLER. Foundations of Optimization. Prentice-Hall, Englewood Cliffs, N.J., 1967.
14. L. A. ZADEH AND C. A. DESOER. "Linear System Theory." McGraw-Hill, New York, 1963.