SIAM J. APPL. MATH.
Vol. 15, No. 3, May, 1967
Printed in U.S.A.

FINITE-STATE PROCESSES AND DYNAMIC PROGRAMMING*

RICHARD M. KARP† AND MICHAEL HELD‡

Abstract. This paper develops a formalism within which the application of dynamic programming to discrete, deterministic problems is rigorously studied. The two central concepts underlying this development are *discrete decision process* and *sequential decision process*. Discrete decision processes provide a convenient means of problem statement, while monotone sequential decision processes (which are finite automata with a certain cost structure superimposed) correspond naturally to dynamic programming algorithms. The representations of discrete decision processes by monotone sequential decision processes are characterized, and this characterization is used in the deviation of dynamic programming algorithms for a variety of problems.

1. Introduction. Dynamic programming is an important technique for the solution of problems involving the optimization of a sequence of decisions. The simple idea underlying this technique is to represent the problem by a process which evolves from state to state in response to decisions. A typical optimization problem then becomes one of guiding the system to a terminal state at minimum cost. When the cost structure is appropriate, the determination of an optimal policy (sequence of decisions) may be reduced to the solution of a functional equation in which the state appears as an independent variable. Bellman's "Principle of Optimality" (cf. [3]), which was the first attempt to formulate this idea in general, was not stated in a form amenable to precise mathematical treatment. Subsequently, several efforts have been directed toward putting the theory on a more rigorous basis. Mitten [13], Denardo [7], and Nemhauser [14] made explicit the role of a basic monotonicity property of cost functions in justifying the functional equations of dynamic programming. Such a monotonicity property is also of importance in the present study. Blackwell [4] and Denardo [7] showed that, with certain assumptions, there is an optimal policy in which each decision taken depends only on the present state. Schreider [17], using a formalism in which a policy is an input string to an automaton (such a formalism is also used in the present paper), characterized optimal policies by functional equations and studied questions of existence and uniqueness for the case in which the state set is compact.

In all of these efforts, however, the representation of the problem in

* Received by the editors May 4, 1966, and in revised form December 12, 1966.

† IBM Thomas J. Watson Research Center, Yorktown Heights, New York. Part of this author's work was supported by Contract AF 30(602)-3546 at the University of Michigan.

‡ IBM Systems Research Institute, 787 United Nations Plaza, New York, New York.

terms of states and transitions is given ab initio. In many applications, however, the state-transition representation is not the most natural form of problem statement, and an appropriate specification of states and transitions is by no means obvious. The recognition of this difficulty was the starting point for the present paper. Thus, this paper develops a theory which, in the case of discrete, deterministic problems, permits state-transition representations and the corresponding functional equations to be derived systematically from a more primitive form of problem statement.

The two central concepts underlying this development are *discrete decision process* and *sequential decision process*. The first of these concepts is introduced to provide a standard format for stating certain optimization problems; the main ingredients of a discrete decision process are a set of primitive decisions, a set of strings of decisions called policies, and a cost function on policies. On the other hand, a sequential decision process is more highly structured; such a process evolves by means of transitions and incremental cost accumulation in response to decisions. Thus, discrete decision processes and sequential decision processes bear the same relation to each other as do the input-output behavior and internal structure of "black boxes", automata, or systems.

Associated with each sequential decision process is a unique discrete decision process. If the sequential decision process has a certain monotonicity property, then a system of recurrence relations can be given whose solution determines the minimum-cost policies for the associated discrete decision process. In view of this, the primary aim of this paper is a characterization (Theorem 2) of the representations of a given discrete decision process by monotone sequential decision processes.

It is our belief that all dynamic programming algorithms for discrete, deterministic optimization problems can be obtained from such representations. This belief is supported in the paper by the systematic (and nearly automatic) derivation of dynamic programming algorithms for a variety of problems.

2. Decision processes. The type of situation we wish to consider is illustrated by the shortest-route problem. This problem is specified by a directed graph with vertex set $V = \{1, 2, \cdots, n\}$, a set of directed edges $E \subseteq V \times V$ and, for each edge (i, j), a positive real number c_{ij} called the *length* of (i, j). It is required to determine a path of minimum total length from vertex 1 to vertex n. We shall assume for uniformity that $E = V \times V$; for otherwise E may be augmented without changing the problem by the insertion of new edges of sufficiently great length.

The usual dynamic programming formulation of this problem is as

follows: for each $j \in V$, let $C(j)$ denote the length of a shortest path from vertex 1 to vertex j. Then, the following system of equations evidently holds:

$$(1) \qquad \begin{aligned} C(1) &= 0, \\ C(j) &= \min_i [C(i) + c_{ij}], \qquad j = 2, 3, \cdots, n. \end{aligned}$$

Various methods can be given for finding the unique solution of this system [1], [15]; once the solution is known, a shortest path from 1 to n is easily found.

The above technique for treating the shortest-route problem is an example of dynamic programming. The purpose of the present section is to define and study a formalization of dynamic programming as it applies to discrete deterministic problems. This development provides a theory which can be used in the construction of dynamic programming algorithms or the proof of their nonexistence. It can often be used to determine that dynamic programming algorithm which, in a certain sense, is the best possible for a given problem.

Discrete dynamic programming involves decisions and distinguished sequences of decisions called policies. In our formal development decisions will be represented as symbols from a finite alphabet, and certain sequences of symbols (words) will be designated as policies. The simplest formalism that has been introduced to define particular sets of words over an alphabet is the finite automaton. Thus finite automata will play an important role in our development, and we begin by discussing them briefly. For more details the reader is referred to [9] and [16].

2.1. Finite automata. A *finite automaton* is a quintuple $\mathfrak{A} = (A, Q, q_0, F, \lambda)$, where A, the *input alphabet*, is a finite nonempty set of *input symbols*, Q is a finite nonempty set of *states*, $q_0 \in Q$ is the *initial state*, $F \subset Q$ is the set of *final states*, and λ, the *transition function*, is a function from $Q \times A$ into Q. The automaton may be viewed as beginning its operation in the initial state q_0, receiving a sequence of input symbols, and executing transitions from state to state in a manner determined by the transition function λ.

Let A^* denote the set of all finite sequences of input symbols; in particular, let the null sequence be denoted e, where $e \notin A$. The elements of A^* are sometimes referred to as *words* over the alphabet A. Let the number of occurrences of input symbols in the word x be called the *length* of x, denoted $\lg(x)$; thus $\lg(e) = 0$. If x and y are elements of A^*, let $x \cdot y$, or simply xy, denote the *concatenation* of the sequences x and y; for example, if $x = a\,c\,b$ and $y = d\,a\,c$, then $xy = a\,c\,b\,d\,a\,c$. In particular, for all

$x \in A^*$, $xe = x$. It is evident that the operation of concatenation is associative, and that A^* is closed under concatenation.

The domain of the transition function λ may be extended to $Q \times A^*$ by the following recursive definition:

$$\lambda(q, e) = q \qquad \text{for all} \quad q \in Q,$$
$$\lambda(q, xa) = \lambda(\lambda(q, x), a) \qquad \text{for all} \quad q \in Q, \quad x \in A^* \text{ and } a \in A.$$

Thus $\lambda(q, x)$ is the state that is reached if the input sequence x is applied, starting in state q. The function $rp(x)$ is defined by the equation $rp(x) = \lambda(q_0, x)$. Thus, $rp(x)$ gives the state that is reached if the input sequence x is applied, starting with $\mathfrak{A}$ in its initial state. The following identities are easily seen to hold:

$$\lambda(q, xy) = \lambda(\lambda(q, x), y) \quad \text{for} \quad x \in A^*, \quad y \in A^*, \tag{2}$$
$$rp(xy) = \lambda(rp(x), y).$$

The word x is said to be *accepted* by $\mathfrak{A}$ if and only if $rp(x) \in F$, and the automaton $\mathfrak{A}$ is said to *define* the set $\{x \mid rp(x) \in F\}$. A set $B \subseteq A^*$ is *regular* if there exists a finite automaton that defines it.

2.2. Discrete decision processes. A *discrete decision process* is a quadruple $\mathfrak{D} = (A, S, P, f)$. The definitions of the elements of $\mathfrak{D}$ are given below with their interpretations adjoined in parentheses:

A is a finite alphabet (the primitive decisions associated with the process);

S is a subset of A^* (the set of policies for the process);

P is an arbitrary set (the set of data specifications for the process);

f is a function (the objective function for the process) from $S \times P$ to R, where R denotes the real numbers.

The *minimization problem* for $\mathfrak{D}$ is as follows: construct an algorithm which, given any $p \in P$, minimizes $f(x, p)$ over all $x \in S$. Thus, we assume that the data specification affects the value of the objective function, but not the set of policies.

Let us consider the interpretation of the shortest-route problem as a discrete decision process; in this discussion, n, the number of vertices, is fixed. Let $A = \{a_1, \cdots, a_n\}$, where a_j has the interpretation "go from the present vertex to vertex j". S consists of all sequences of the form xa_n, $x \in A^*$, P is the set of all $n \times n$ real positive matrices (c_{kl}), and f is determined as follows: if $x = a_{i_1}a_{i_2} \cdots a_{i_r}a_n$, then $f(x, (c_{kl})) = c_{1i_1} + c_{i_1i_2} + \cdots + c_{i_rn}$. The shortest-route problem is precisely the following minimization problem: given (c_{kl}), minimize $f(x, (c_{kl}))$ over all $x \in S$.

The dynamic programming approach to solving the minimization

problem for a discrete decision process rests on a certain type of representation of the process, which we now proceed to specify.

2.3. Sequential decision processes. A sequential decision process is a finite automaton upon which a certain cost structure is superimposed. Formally, a *sequential decision process* II is specified by:

(i) a finite automaton $\mathfrak{A} = (A, Q, q_0, F, \lambda)$,

(ii) a set P,

(iii) a function $h: R \times Q \times A \times P \to R$,

(iv) a function $k: P \to R$.

The real-valued functions h and k are used to introduce the notion of "cost" into the sequential decision process. For a given data specification $p \in P$, $k(p)$ gives the cost of the null sequence e. The quantity $h(\xi, q, a, p)$ gives the cost of reaching the state $\lambda(q, a)$ by an input sequence that reaches q at cost ξ (where $\xi \in R$) and is then extended by the input a. In line with this interpretation h may be extended to associate costs with input sequences $x \in A^*$ according to the following recursive definition:

$$h(\xi, q, e, p) = \xi,$$

$$h(\xi, q, xa, p) = h(h(\xi, q, x, p), \lambda(q, x), a, p).$$

Of particular importance is the quantity $h(k(p), q_0, x, p)$, which gives the cost of starting in the initial state at cost $k(p)$, and then applying the input sequence x. Thus, for convenience, we define a function $g(x, p)$ by the equation

$$g(x, p) = h(k(p), q_0, x, p).$$

The following identities are immediate:

$$h(\xi, q, xy, p) = h(h(\xi, q, x, p), \lambda(q, x), y, p),$$

$$g(e, p) = k(p),$$

$$g(xy, p) = h(g(x, p), rp(x), y, p).$$

Thus, in a sequential decision process, the cost of an input sequence is accumulated step-by-step, whereas, in a discrete decision process, the cost is presented in an arbitrary manner.

The *minimization problem* for a sequential decision process II is as follows: construct an algorithm which, given any $p \in P$, minimizes $g(x, p)$ over the set of all words x accepted by the automaton $\mathfrak{A}$.

The sequential decision process $\text{II} = (\mathfrak{A}(A, Q, q_0, F, \lambda), P, h, k)$ is said to *represent* the discrete decision process $\mathfrak{D} = (A, S, P, f)$ if $S \subseteq A^*$ is the set of words accepted by $\mathfrak{A}$ and, for all $(x, p) \in S \times P$, $f(x, p) = g(x, p)$. For example, the n-city shortest-route problem is represented by the follow-

ing sequential decision process: $A = \{a_1, a_2, \cdots, a_n\}$; $Q = \{q_1, q_2, \cdots, q_n\}$; the initial state is q_1; $F = \{q_n\}$; for all i and j, $\lambda(q_i, a_j) = q_j$; P is the set of all positive $n \times n$ real matrices (c_{kl}); $h(\xi, q_i, a_j, (c_{kl})) = \xi + c_{ij}$; $k(p) = 0$ for all $p \in P$.

Clearly, any sequential decision process II represents a unique discrete decision process $\mathfrak{D}$, and the minimization problems for II and $\mathfrak{D}$ are equivalent. We shall be interested in the converse problem of finding a sequential decision process II to represent a given discrete decision process $\mathfrak{D}$.

2.4. Monotone sequential decision processes. We next introduce a class of sequential decision processes with a certain monotonicity property, for which the minimization problem is particularly simple, since it reduces to the solution of a system of recurrence equations of the type normally associated with dynamic programming. The sequential decision process II is said to be *monotone* if, for all $(q, a, p) \in Q \times A \times P$, the following implication holds:

$$\xi_1 \leqq \xi_2 \quad \text{implies} \quad h(\xi_1, q, a, p) \leqq h(\xi_2, q, a, p).$$

The property of monotonicity may be extended to the domain $Q \times A^* \times P$, as is easily seen by induction on $\lg(x)$:

$$(3) \quad \xi_1 \leqq \xi_2 \quad \text{implies} \quad h(\xi_1, q, x, p) \leqq h(\xi_2, q, x, p), \qquad x \in A^*.$$

As a particular case of (3), we obtain the following:

$$(4) \quad \begin{aligned} &\text{if} \quad rp(x) = rp(y) \quad \text{and} \quad g(x, p) \leqq g(y, p), \\ &\qquad\qquad \text{then} \quad g(xw, p) \leqq g(yw, p) \quad \text{for all } w. \end{aligned}$$

LEMMA 1. *Let* II *be a monotone sequential decision process, and let* S *be the set of words accepted by* $\mathfrak{A}$. *For* $p \in P$, *let* z *be called optimal with respect to* p *if* $z \in S$ *and* $g(z, p) = \min_{y \in S} g(y, p)$. *Let* v *and* w *be words such that* $rp(v) = rp(w)$ *and* $g(v, p) \leqq g(w, p)$. *Then, if* wx *is optimal with respect to* p, vx *is also optimal with respect to* p.

Proof. Assume wx is optimal with respect to p. Since $rp(v) = rp(w)$, $rp(vx) = rp(wx)$. Thus $vx \in S$, since $wx \in S$. Also, since $rp(v) = rp(w)$ and $g(v, p) \leqq g(w, p)$, it follows from (4) that $g(vx, p) \leqq g(wx, p)$. Also, by the optimality of wx, $g(wx, p) \leqq g(vx, p)$. Thus, $g(vx, p) = g(wx, p) = \min_{y \in S} g(y, p)$, and vx is optimal with respect to p. This completes the proof.

Loosely speaking, Lemma 1 establishes that, if there is an optimal policy which reaches an intermediate state q, then there is an optimal policy which reaches q at minimum cost. This result is similar to the principle of optimality [3, p. 83].

Now, for $q \in Q$, let[1]

$$G(q, p) = \min_{\{x \mid rp(x)=q\}} g(x, p);$$

thus $G(q, p)$ gives the minimum cost of reaching state q from the initial state. The following theorem establishes the validity of the functional equations characteristic of dynamic programming.

THEOREM 1. *Let* Π *be a monotone sequential decision process. Then, for any* $p \in P$, *the following equations hold*:

$$\begin{aligned} G(q_0, p) &= \min[g(e, p), \min_{\{(q',a) \mid \lambda(q',a)=q_0\}} (h(G(q', p), q', a, p))]; \\ G(q, p) &= \min_{\{(q',a) \mid \lambda(q',a)=q\}} [h(G(q', p), q', a, p)] \quad \text{for} \quad q \neq q_0. \end{aligned} \tag{5}$$

Proof. For each $q \in Q$, let the right-hand side of the equation for $G(q, p)$ be denoted $H(q, p)$. First, we shall show that $G(q, p) \geqq H(q, p)$. By definition, $G(q, p) = g(x, p)$ for some $x \in A^*$ such that $rp(x) = q$. If $x = e$, so that $q = rp(e) = q_0$, then $G(q_0, p) = g(e, p)$, and, by inspection of (5), $g(e, p) \geqq H(q_0, p)$. If $x \neq e$, then x is of the form ya, where $a \in A$. Then $G(q, p) = g(ya, p) = h(g(y, p), rp(y), a, p)$. But, by the definition of G, $g(y, p) \geqq G(rp(y), p)$ and, since h is monotone, $h(g(y, p), rp(y), a, p) \geqq h(G(rp(y), p), rp(y), a, p)$. But, from (5) it follows that $h(G(rp(y), p), rp(y), a, p) \geqq H(q, p)$. Tracing back through the inequalities that have been derived, we see that, in the case $x \neq e$, $G(q, p) \geqq H(q, p)$. To complete the proof, we must show that $G(q, p) \leqq H(q, p)$. This is certainly true if $q = q_0$ and $H(q_0, p) = g(e, p)$. Suppose that $H(q, p) = h(G(q', p), q', a, p)$, where $\lambda(q', a) = q$. By definition, $G(q', p) = g(y, p)$ for some y such that $rp(y) = q'$. Then $rp(ya) = q$, and $G(q, p) \leqq g(ya, p) = h(g(y, p), q', a, p) = h(G(q', p), q', a, p) = H(q, p)$. This completes the proof.

As a typical application of Theorem 1 we note that, since the sequential decision process representing the shortest-route problem is monotone,

$$\begin{aligned} G(q_1, (c_{kl})) &= \min\{0, \min_i [G(q_i, (c_{kl})) + c_{i1}]\}, \\ G(q_j, (c_{kl})) &= \min_i [G(q_i, (c_{kl})) + c_{ij}], \qquad j \neq 1. \end{aligned}$$

Since the function h is positive in this case, the above equations, with a minor change in notation, reduce to the system of equations (1).

Using Theorem 1 it is possible to define an approach to the minimization problem for discrete decision processes. Given such a process $\mathfrak{D}$, the first

[1] We shall be interested only in those cases where this minimum exists. This will be the case, for example, when it is unprofitable to traverse a cycle through the states more than a specified number of times.

step is to represent it by a monotone sequential decision process Π and derive the system of equations (5). The second step is to give a method of solving (5) for any $p \in P$, and of deriving from that solution a word which is optimal with respect to p. Questions of existence, uniqueness, and methods of solution for (5) are presently understood only for some important special cases (cf. [15], for example). Our primary concern in the present paper is to characterize the representations of discrete decision processes by monotone sequential decision processes. We turn next to this problem.

3. The representation problem. In order to characterize the representations of discrete decision processes by monotone sequential decision processes, some additional background material from automata theory will be required. The main concept introduced will be that of the equiresponse relation of a finite automaton $\mathfrak{A}$ (cf. [9]). The *equiresponse relation* of $\mathfrak{A}$ is an equivalence relation $\sim$ over A^*, defined as follows: $x \sim y$ if and only if $rp(x) = rp(y)$. By extension, if $\Pi = (\mathfrak{A}, P, h, k)$ is a sequential decision process and $\sim$ is the equiresponse relation of $\mathfrak{A}$, then we also refer to $\sim$ as the equiresponse relation of Π. Any equivalence relation $\equiv$ over A^* is called a *right congruence* if $x \equiv y$ implies for all $w \in A^*$, $xw \equiv yw$. Then it follows easily from the identity $rp(xw) = \lambda(rp(x), w)$ that any equiresponse relation $\sim$ is a right congruence; also, since the number of states is finite, $\sim$ is of finite rank (i.e., it has a finite number of equivalence classes).

Let δ and Δ be equivalence relations over A^*. Then δ is said to *refine* Δ (written $\delta \leqq \Delta$) if, for all $x, y \in A^*$, $x\delta y$ implies $x\Delta y$. For any $S \subseteq A^*$, let the right congruence R_S be defined as follows:

$$x R_S y \quad \text{if and only if for all} \quad v \in A^*, \quad xv \in S \quad \text{if and only if} \quad yv \in S.$$

Thus an equivalence class of R_S is a maximal set of words which "act alike", with respect to membership in S, under all continuations.

The following well-known lemma [9], [16] is basic to the study of finite automata.

LEMMA 2. *Let S be a subset of A^* and let $\sim$ be a right congruence of finite rank. Then $\sim$ is the equiresponse relation of some finite automaton defining S if and only if $\sim$ refines R_S.*

Proof. Let $\sim$ be the equiresponse relation of a finite automaton $\mathfrak{A}$ defining S. Then, as we have already remarked, $\sim$ is a right congruence of finite rank. Also, if $x \sim y$ then $rp(x) = rp(y)$, and, for any v, $rp(xv) = rp(yv)$. Since $xv \in S$ if and only if $rp(xv) \in F$, it follows that $xv \in S$ if and only if $yv \in S$, so that $x R_S y$. Thus $x \sim y$ implies $x R_S y$, so that $\sim$ refines R_S. Conversely, let $\sim$ be a right congruence of finite rank over A^* refining R_S. We shall specify a finite automaton $\mathfrak{A}$ defining S and having

$\sim$ as its equiresponse relation. For $x \in A^*$, let $[x]$ denote the equivalence class of x in $\sim$. Then

$$Q = \{[x] \mid x \in A^*\},$$
$$q_0 = [e],$$
$$F = \{[x] \mid x \in S\},$$
$$\lambda([x], a) = [xa].$$

Because $\sim$ is a right congruence, λ is well defined. Because $\sim$ is of finite rank, $\mathfrak{A}$ has only a finite number of states. By induction on $\lg(x)$ it follows that, for all $x \in A^*$, $rp(x) = [x]$, so that the equiresponse relation of $\mathfrak{A}$ is indeed $\sim$. Because $\sim$ refines R_S, $x \in S$ if and only if every element of the equivalence class $[x]$ is in S; thus, F is well defined, and with the given definition, the set of words accepted by $\mathfrak{A}$ is S. This completes the proof.

The construction used above to obtain a finite automaton having a given equiresponse relation $\sim$ and defining a given set S will be referred to as the *standard construction.*

The following theorem characterizes the representations of discrete decision processes by monotone sequential decision processes. This theorem, which is the main result of the present paper, provides a systematic approach to the construction of dynamic programming algorithms, and will be applied throughout the remainder of the paper.

THEOREM 2. *Let* $\mathfrak{D} = (A, S, P, f)$ *be a discrete decision process, and let* $\sim$ *be an equivalence relation over* A^*. *Then there exists a monotone sequential decision process* $\Pi = (\mathfrak{A}, P, h, k)$ *such that* Π *represents* $\mathfrak{D}$ *and has* $\sim$ *as its equiresponse relation if and only if the following conditions are satisfied*:

(i) $\sim$ *is a right congruence of finite rank which refines* R_S;

(ii) *if* $x \sim y$, *then, for each* $p \in P$, *either* (a) *or* (b) *holds*:

(a) *for all* w *such that* $xw \in S$, $f(xw, p) \leqq f(yw, p)$,

(b) *for all* w *such that* $xw \in S$, $f(xw, p) \geqq f(yw, p)$;

(iii) *if* $x \sim y$, $x \in S$, $y \in S$, *and* $f(x, p) = f(y, p)$, *then, for all* w *such that* $xw \in S$ *and* $yw \in S$, $f(xw, p) = f(yw, p)$.

Before giving the proof we note that conditions (i) and (iii) are together necessary and sufficient in order that $\sim$ be the equiresponse relation of a (not necessarily monotone) sequential decision process representing $\mathfrak{D}$. The key additional condition required to characterize *monotone* representations is (ii), which expresses a certain comparability between x and y with regard to the costs of their possible continuations.

Proof of Theorem 2. Suppose that $\mathfrak{D}$ is represented by Π, and consider the equiresponse relation $\sim$ of Π. By Lemma 2, condition (i) holds. Since Π represents $\mathfrak{D}$, it follows that, if $x \sim y$ and $xw \in S$, then $g(xw, p)$

$= f(xw, p)$, and $g(yw, p) = f(yw, p)$. Thus if $g(x, p) \leqq g(y, p)$, the inequality (4) implies condition (ii)(a); similarly if $g(x, p) \geqq g(y, p)$, condition (ii) (b) holds. Finally, condition (iii) follows easily. For, if $x \sim y$, or equivalently, $rp(x) = rp(y)$, then $f(x, p) = f(y, p)$ implies $g(x, p) = g(y, p)$, which implies, for all w, $g(xw, p) = g(yw, p)$, which in turn implies that for all w such that $xw \in S$, $f(xw, p) = f(yw, p)$.

To complete the proof, we shall show that, if $\sim$ satisfies the given conditions, there exists a representation of $\mathfrak{D} = (A, S, P, f)$ by a monotone sequential decision process $\Pi = (\mathfrak{A}, P, h, k)$ such that $\mathfrak{A} = (A, Q, q_0, F, \lambda)$ has $\sim$ as its equiresponse relation. The construction of the required automaton $\mathfrak{A}$, defining the set S, is carried out by the standard construction given in the proof of Lemma 2. Since condition (i) holds, Lemma 2 guarantees that this is possible. It remains only to specify the functions h and k; this will be done by first specifying the function g. For $q \in Q$, let $X(q) = \{x \mid rp(x) = q\}$. For each $p \in P$, a relation $\preceq_{p,q}$ over $X(q)$ may be defined as follows: $x \preceq_{p,q} y$ if and only if (ii)(a) holds. Since (ii) holds, it follows that either $x \preceq_{p,q} y$ or $y \preceq_{p,q} x$. It is possible, even for distinct words x and y, that $x \preceq_{p,q} y$ and $y \preceq_{p,q} x$; this is the case if and only if, for all w such that $xw \in S$, $f(xw, p) = f(yw, p)$. Let us define an equivalence relation $=_{p,q}$ by the condition $x =_{p,q} y$ if and only if $x \preceq_{p,q} y$ and $y \preceq_{p,q} x$. Let the equivalence class of x under the relation $=_{p,q}$ be denoted $\langle x \rangle_{p,q}$. Then the relation $\preceq_{p,q}$ is inherited by the set of equivalence classes, and is a total ordering of these classes. Therefore, there exists a function $\gamma_{p,q}$ from the set of equivalence classes into the reals such that

$$\gamma_{p,q}(\langle x \rangle_{p,q}) < \gamma_{p,q}(\langle y \rangle_{p,q}) \quad \text{if and only if} \quad \langle x \rangle_{p,q} \preceq_{p,q} \langle y \rangle_{p,q} \text{ and } \langle x \rangle_{p,q} \neq \langle y \rangle_{p,q}. \tag{6}$$

In particular, when $q \in F$, the following specification of γ will be adopted:

$$\gamma_{p,q}(\langle x \rangle_{p,q}) = f(x, p).$$

We shall show that this specification of $\gamma_{p,q}$ satisfies (6). Since $\gamma_{p,q}(\langle x \rangle_{p,q}) = f(x, p)$, (6) reduces to:

$$f(x, p) < f(y, p) \quad \text{if and only if} \quad \langle x \rangle_{p,q} \preceq_{p,q} \langle y \rangle_{p,q} \text{ and } \langle x \rangle_{p,q} \neq \langle y \rangle_{p,q}.$$

Given that $f(x, p) < f(y, p)$ condition (ii) ensures the desired conclusion. Conversely, $\langle x \rangle_{p,q} \preceq_{p,q} \langle y \rangle_{p,q}$ implies, by definition, that, when $xw \in S$, $f(xw, p) \leqq f(yw, p)$ and, in particular, that $f(x, p) \leqq f(y, p)$. Also, by condition (iii), $f(x, p) = f(y, p)$ implies $\langle x \rangle_{p,q} = \langle y \rangle_{p,q}$, contradicting the hypothesis that $\langle x \rangle_{p,q} \neq \langle y \rangle_{p,q}$. Thus, $f(x, p) \leqq f(y, p)$, and (6) is satisfied. Now, let the function $g(x, p)$ be defined as follows: for all q and all x

$\in X(q)$, $g(x, p) = \gamma_{p,q}(\langle x\rangle_{p,q})$. With this specification, it follows immediately that, for all $x \in S$, $g(x, p) = f(x, p)$. Having defined g, we can easily specify the functions k and h. The function k is determined by the equation $k(p) = g(e, p) = \gamma_{p,q_0}(\langle e\rangle_{p,q_0})$. The specification of g also determines the function $h(\xi, q, a, p)$ for all ξ such that $\xi = g(x, p)$ for some $x \in X(q)$, as follows: $h(g(x, p), q, a, p) = g(xa, p)$, or, equivalently, $h(\gamma_{p,q}(\langle x\rangle_{p,q}), q, a, p) = \gamma_{p,q'}(\langle xa\rangle_{p,q'})$, where $q' = \lambda(q, a)$. This partial specification of h does not violate the requirement that h be monotone, since

$$\begin{aligned}\gamma_{p,q}(\langle x\rangle_{p,q}) \leqq \gamma_{p,q}(\langle y\rangle_{p,q}) &\Rightarrow \langle x\rangle_{p,q} \preceq_{p,q} \langle y\rangle_{p,q}\\ &\Rightarrow \langle xa\rangle_{p,q'} \preceq_{p,q'} \langle ya\rangle_{p,q'}\\ &\Rightarrow \gamma_{p,q'}(\langle xa\rangle_{p,q'}) \leqq \gamma_{p,q'}(\langle ya\rangle_{p,q'}).\end{aligned}$$

By suitable interpolation (piecewise-linear, for example), the function h can be extended, in a monotonicity-preserving fashion, so that it is defined for all real ξ. This completes the proof of Theorem 2.

It will be convenient to make note of the following corollary, whose proof requires a modification of the construction used in the proof of Theorem 2.

COROLLARY 1. *Let $\mathfrak{D} = (A, S, P, f)$ be a discrete decision process, let $\sim$ be an equivalence relation satisfying the hypotheses of Theorem 2, and let $s(x, p)$ be a function from $A^* \times P$ into the reals. Then the following conditions are equivalent*:

(a) *The function $s(x, p)$ satisfies*:
  (i) *if $x \in S$, then $s(x, p) = f(x, p)$*;
  (ii) *if $x \sim y$, $f(xw, p) \leqq f(yw, p)$ for all w such that $xw \in S$, and, for some such w, $f(xw, p) < f(yw, p)$, then $s(x, p) < s(y, p)$*;
  (iii) *if $s(x, p) < s(y, p)$, then, for all w, $s(xw, p) \leqq s(yw, p)$.*

(b) *There exists a monotone sequential decision process Π representing $\mathfrak{D}$ having $\sim$ as its equiresponse relation and such that the function $g(x, p)$ associated with Π is identically equal to $s(x, p)$.*

It is a direct consequence of Lemma 2 that a set $S \subseteq A^*$ is regular (i.e., defined by some finite automaton) if and only if the right congruence R_S is of finite rank. Thus, a discrete decision process $\mathfrak{D} = (A, S, P, f)$ where R_S is not of finite rank cannot be represented by a sequential decision process. Even if S is regular, $\mathfrak{D}$ may not be representable by a monotone sequential decision process. For example, let $A = \{0, 1\}$, let S consist of all sequences of the form 0^n1, $n \geqq 0$, and let the cost function be specified, independent of parameters, as follows:

$$f(0^n1) = \begin{cases} +1 & \text{if } n \text{ is a perfect square,}\\ 0 & \text{otherwise.}\end{cases}$$

It is easily verified that S is regular, and, in fact, that R_S has the following

three equivalence classes:

(1) all sequences of the form 0^n;

(2) all sequences of the form 0^n1;

(3) all other sequences.

However there is no equivalence relation $\sim$ of finite rank satisfying condition (ii) of Theorem 2 in this case; for it is easily shown that, given any two nonnegative integers k and l, there exist integers m and n such that $k + m$ and $l + n$ are perfect squares, but $k + n$ and $l + m$ are not. Therefore $f(0^k0^m1) > f(0^l0^m1)$, but $f(0^k0^n1) < f(0^l0^n1)$; and hence, if $0^k \sim 0^l$, then (ii) is not satisfied. Thus, for arbitrary nonnegative integers k and l, it cannot be the case that $0^k \sim 0^l$, and $\sim$ cannot be of finite rank.

A slight modification of this example can be constructed in which, although S is regular, the choice of any right congruence $\sim$ of finite rank refining R_S would cause the violation of condition (iii) of Theorem 2. Thus the discrete decision process of such an example is not representable by any sequential decision process, monotone or not.

4. A module placement problem. The preceding concepts, particularly Theorem 2, can be used to construct dynamic programming algorithms or to demonstrate their nonexistence. As an illustration let us consider the following simple module placement problem. Suppose we are given n positions, indexed $1, 2, \cdots, n$, 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. Then it is desired to 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 $|i - j|$.

We shall consider two alternate formulations of this problem as a discrete decision process. In the first formulation, we let $\Pi(i)$ denote the position in which object i is placed. Then the decision process $D^{(1)} = (A^{(1)}, S^{(1)}, P^{(1)}, f^{(1)})$ is specified as follows: $A^{(1)} = \{1, 2, \cdots, n\}$; the element $j \in A^{(1)}$ has the interpretation "place the next object in position j"; $S^{(1)}$ is the set of all sequences $\Pi(1)\ \Pi(2) \cdots \Pi(n)$ such that $\Pi(\cdot)$ is a permutation; $P^{(1)}$ is the set of all $n \times n$ symmetric nonnegative integer matrices (a_{ij}); and

$$f^{(1)}(\Pi(1) \cdots \Pi(n), (a_{ij})) = \sum_{\{i,j \mid i<j\}} a_{ij} |\Pi(i) - \Pi(j)|.$$

It can be verified that the induced right congruence $R_{S^{(1)}}$ has $2^n + 1$ equivalence classes. One of these classes consists of all the elements of $A^{(1)*}$ in which some index occurs more than once. The other equivalence classes are in one-to-one correspondence with the 2^n subsets of $A^{(1)}$. The equivalence class corresponding to the subset $T = \{i_1, i_2, \cdots, i_r\} \subseteq A^{(1)}$

consists of all sequences in which each element of T occurs exactly once, and no other elements occur. Each member of this equivalence class determines an assignment of the first r objects to the positions in the set T. Using Theorem 2 we shall show that there is no monotone sequential decision process representing $\mathfrak{D}^{(1)}$ having $R_{S(1)}$ as its equiresponse relation; specifically, we shall show that $R_{S(1)}$ does not satisfy condition (ii) of Theorem 2. We begin by noting the following symmetry property, which results from the structure of the function $f^{(1)}$. Given a permutation Π, its *reversal* Π^R is defined by the equation $\Pi^R(i) = n + 1 - \Pi(i)$, $i = 1, 2, \cdots, n$. Note that $(\Pi^R)^R = \Pi$. Then it is easy to check that

$$(7)\quad f^{(1)}(\Pi(1) \cdots \Pi(n), (a_{ij})) = f^{(1)}(\Pi^R(1)\Pi^R(2) \cdots \Pi^R(n), (a_{ij})).$$

Now consider the equivalence class corresponding to the set $\{1, n\}$; its two elements are the sequences $(1, n)$ and $(n, 1)$. If $R_{S(1)}$ satisfied condition (ii) of Theorem 2, it would be the case that, for every data specification (a_{ij}), one of the following two statements must hold:

(a) for all permutations Π such that $\Pi(1) = 1$ and $\Pi(2) = n$,

$$f^{(1)}(1n\Pi(3) \cdots \Pi(n), (a_{ij})) \leqq f^{(1)}(n1\Pi(3) \cdots \Pi(n), (a_{ij}));$$

(b) for all permutations Π such that $\Pi(1) = 1$ and $\Pi(2) = n$,

$$f^{(1)}(1n\Pi(3) \cdots \Pi(n), (a_{ij})) \geqq f^{(1)}(n1\Pi(3) \cdots \Pi(n), (a_{ij})).$$

Suppose, for example, that the former condition holds. Then, for any permutation Π of the type defined above, the following inequalities hold:

$$f^{(1)}(1n\Pi(3) \cdots \Pi(n), (a_{ij})) \leqq f^{(1)}(n1\Pi(3) \cdots \Pi(n), (a_{ij})),$$

$$f^{(1)}(1n\Pi^R(3) \cdots \Pi^R(n), (a_{ij})) \leqq f^{(1)}(n1\Pi^R(3) \cdots \Pi^R(n), (a_{ij})).$$

But, applying (7) to the second of these inequalities,

$$f^{(1)}(n1\Pi(3) \cdots \Pi(n), (a_{ij})) \leqq f^{(1)}(1n\Pi(3) \cdots \Pi(n), (a_{ij})).$$

It follows that

$$f^{(1)}(1n\Pi(3) \cdots \Pi(n), (a_{ij})) = f^{(1)}(n1\Pi(3) \cdots \Pi(n), (a_{ij})).$$

The same conclusion results from the assumption that condition (b) holds. But clearly this equality will be violated by almost all choices of the data specifications (a_{ij}).

Thus the discrete decision process $\mathfrak{D}^{(1)}$ cannot be represented by a monotone sequential decision process with only $2^n + 1$ states, and does not yield a $(2^n + 1)$-state dynamic programming formulation of the module placement problem. A second formulation of this problem as a discrete decision process will, however, yield such an algorithm. In this second formulation,

we let $\rho(i)$ denote the object placed in position i, and specify the discrete decision process $D^{(2)} = (A^{(2)}, S^{(2)}, P^{(2)}, f^{(2)})$ as follows: $A^{(2)} = \{1, 2, \cdots, n\}$, where, this time, the element $j \in A^{(2)}$ has the interpretation "fill the next position with object j"; $S^{(2)}$ is the set of all sequences $\rho(1)\rho(2) \cdots \rho(n)$ such that $\rho(\cdot)$ is a permutation; $P^{(2)}$ is the set of all $n \times n$ symmetric nonnegative integer matrices (a_{ij});

$$f^{(2)}(\rho(1) \cdots \rho(n), (a_{ij})) = \sum_{\{i,j \mid i<j\}} a_{\rho(i),\rho(j)}(j - i).$$

The induced right congruence $R_{S^{(2)}}$ has $2^n + 1$ equivalence classes. It has exactly the same mathematical structure as the right congruence $R_{S^{(1)}}$ obtained in the first formulation, but is interpreted differently. In the second formulation each element of the equivalence class corresponding to the set $T = \{i_1, i_2, \cdots, i_r\}$ specifies an assignment of the elements of this set to the first r positions.

We shall show that there is a monotone sequential decision process representing $D^{(2)}$ and having $R_{S^{(2)}}$ as its equiresponse relation. This will be done by showing that the conditions of Theorem 2 are satisfied. It is immediate that condition (i) is satisfied, and condition (iii) is satisfied trivially, in the present case, since $x \in S$ and $xw \in S$ imply $w = e$. Thus it remains only to demonstrate that condition (ii) is satisfied. For the equivalence class consisting of all sequences with repeated elements, this condition is satisfied vacuously. Let $x = \sigma(1)\sigma(2) \cdots \sigma(r)$ and $y = \tau(1)\tau(2) \cdots \tau(r)$ be two elements of the equivalence class corresponding to the set T; then $\{\sigma(1), \sigma(2), \cdots, \sigma(r)\} = \{\tau(1), \tau(2), \cdots, \tau(r)\} = T$. Let $w = \theta(r+1)\theta(r+2) \cdots \theta(n)$ be such that xw and yw are elements of $S^{(2)}$. Thus, each element of $\{1, 2, \cdots, n\}$ occurs exactly once both in xw and in yw, so that, necessarily, $\{\theta(r+1), \theta(r+2), \cdots, \theta(n)\}$ is the complement $\bar{T}$ of T with respect to $\{1, 2, \cdots, n\}$. Then

$$f^{(2)}(xw; (a_{ij})) = \sum_{\substack{i,j=1 \\ i<j}}^{r} a_{\sigma(i),\sigma(j)}(j - i) + \sum_{\substack{i,j=r+1 \\ i<j}}^{n} a_{\theta(i),\theta(j)}(j - i) + \sum_{i=1}^{r} \sum_{j=r+1}^{n} a_{\sigma(i),\theta(j)}(j - i).$$

The last of these three terms may be rewritten as

$$\sum_{i=1}^{r} \sum_{j=r+1}^{n} a_{\sigma(i),\theta(j)}\, j - \sum_{i=1}^{r} \sum_{j=r+1}^{n} a_{\sigma(i),\theta(j)}\, i.$$

Because $\sigma(\cdot)$ ranges over T, and $\theta(\cdot)$, over $\bar{T}$, this becomes

$$\sum_{\substack{j=r+1 \\ l \in T}}^{n} a_{l,\theta(j)}\, j - \sum_{\substack{i=1 \\ m \in \bar{T}}}^{r} a_{\sigma(i),m}\, i.$$

The quantity $f^{(2)}(yw; (a_{ij}))$ can be expressed similarly, and thus

$$f^{(2)}(xw, (a_{ij})) - f^{(2)}(yw, (a_{ij})) = \left(\sum_{\substack{i,j=1 \\ i<j}}^{r} a_{\sigma(i),\sigma(j)}(j - i) - \sum_{\substack{i=1 \\ m \in \bar{T}}}^{r} a_{\sigma(i),m} i \right) - \left(\sum_{\substack{i,j=1 \\ i<j}}^{r} a_{\tau(i),\tau(j)}(j - i) - \sum_{\substack{i=1 \\ m \in \bar{T}}}^{r} a_{\tau(i),m} i \right). \tag{8}$$

But the right-hand side of this expression is the same for all permissible choices of w, and condition (ii) is verified.

Thus, Theorem 2 guarantees that there exists a monotone sequential decision process $\Pi = (\mathfrak{A}, P, k, h)$ representing $\mathfrak{D}^{(2)}$ and having $R_{S^{(2)}}$ as its equiresponse relation.

The construction of one such process Π will now be carried out. The finite automaton $\mathfrak{A}$ is obtained by applying the standard construction. Let names be assigned to the states of $\mathfrak{A}$ according to the following convention: If $x = x_{i_1}x_{i_2} \cdots x_{i_r}$ is an input word containing no repeated symbols, then $rp(x) = \{x_{i_1}, x_{i_2}, \cdots, x_{i_r}\}$. If x contains a repeated symbol, then $rp(x) = \Omega$. Thus, except for the special state Ω, the name of each state is that of a subset of $\{1, 2, \cdots, n\}$. From (8) it is evident that a function $s(x, p)$ satisfying the conditions of Corollary 1 may be specified as follows:

(i) if $x = \sigma(1)\sigma(2) \cdots \sigma(r)$, where $\sigma(\cdot)$ is a permutation and $\{\sigma(1), \sigma(2), \cdots, \sigma(r)\} = T$, then

$$s(x, p) = \sum_{\substack{i,j=1 \\ i<j}}^{r} a_{\sigma(i),\sigma(j)}(j - i) - \sum_{\substack{i=1 \\ m \in \bar{T}}}^{r} a_{\sigma(i),m} i; \tag{9}$$

(ii) if x has a repeated element, then $s(x, p) = s(y, p)$, where y is the longest prefix of x without repetitions.

As the reader will soon see, this apparently arbitrary choice will permit the function h to assume a special additive form. The function $g(x, p)$ associated with Π will be set equal to $s(x, p)$.[2] Further, $k(p) = g(e, p) = 0$. Recalling that the function h satisfies the identity $h(g(x, p), T, a, p) = g(xa, p)$, where $T = \{\sigma(1), \cdots, \sigma(r)\} = rp(x)$, and assuming that h is of the "additive" form $h(\xi, T, a, p) = \xi + I(T, a, p)$, so that $g(xa, p) = g(x, p) + I(T, a, p)$, we obtain, after some manipulation,

$$I(T, \sigma(r + 1), p) = (r + 1) \left(\sum_{i=1}^{r} a_{\sigma(i),\sigma(r+1)} - \sum_{\substack{m \in \bar{T} \\ m \neq \sigma(r+1)}} a_{\sigma(r+1),m} \right).$$

Also, $I(T, a, p) = 0$ if $a \in T$, and $I(\Omega, a, p) = 0$ for all a.

[2] A different, and possibly more intuitive, choice of $g(x, p)$ will be presented in §5.

This completes the specification of Π. Applying Theorem 1, we obtain the following system of recurrence equations, which is the basis of a dynamic programming algorithm for solving the module placement problem:

$$G(\varnothing, p) = 0,$$

$$G(T, p) = \min_{l \in T} [G(T - l, p) + I(T - l, l, p)] \quad \text{for} \quad T \subseteq \{1, 2, \cdots, n\}, \quad T \neq \varnothing,$$

$$G(\Omega, p) = \min [G(\Omega, p), \min_{T \neq \varnothing} G(T, p)].$$

The determination of $G(\Omega, p)$ is, of course, unnecessary since there is no path leading from Ω to the final state $\{1, 2, \cdots, n\}$.

5. Additive processes. In the two examples so far considered, the shortest-route problem and the module placement problem, the cost function h assumed the additive form

$$h(\xi, q, a, p) = \xi + I(q, a, p), \tag{10}$$

where $I(q, a, p)$ may be interpreted as the incremental cost of applying the input a while in the state q. In view of the importance of this cost structure in applications, we shall discuss it in some detail.

A sequential decision process $\Pi = (\mathfrak{A}, P, h, k)$ is said to be *additive* if h is of the form (10). Clearly any additive sequential decision process is monotone. The following theorem characterizes the representation of discrete decision processes by additive sequential decision processes.

THEOREM 3. *Let $\mathfrak{D} = (A, S, P, f)$ be a discrete decision process, and let $\sim$ be an equivalence relation over A^*. Then there exists an additive sequential decision process $\Pi = (\mathfrak{A}, P, h, k)$ such that Π represents $\mathfrak{D}$ and has the equi-response relation $\sim$ if and only if the following conditions hold:*

(i) *$\sim$ is a right congruence of finite rank which refines R_S;*

(ii) *if $x \sim y$, $xw \in S$ and $xv \in S$, then $f(xw, p) - f(yw, p) = f(xv, p) - f(yv, p)$.*

Proof. Let an additive sequential decision process $\Pi = (\mathfrak{A}, P, h, k)$ representing $\mathfrak{D} = (A, S, P, f)$ be given. Since the automaton $\mathfrak{A}$ defines the set S, Lemma 2 guarantees the necessity of (i). To prove the necessity of (ii), suppose that $rp(x) = rp(y) = q$ (i.e., $x \sim y$). Then, by the definition of g and the additivity of h,

$$g(xa, p) - g(x, p) = g(ya, p) - g(y, p) = I(q, a, p).$$

Also, since $rp(xa) = rp(ya)$, an induction on $\log(w)$ may be carried out to yield:

$$g(xw, p) - g(x, p) = g(yw, p) - g(y, p)$$

for all w. Since g agrees with f on S, it follows that, whenever $xw \in S$, $f(xw, p) - f(yw, p) = g(x, p) - g(y, p)$. Similarly, $f(xv, p) - f(yv, p) = g(x, p) - g(y, p)$, and the necessity of (ii) is established.

We now turn to the proof of sufficiency. Given $\mathfrak{D}$ and the relation $\sim$ we shall construct an appropriate II. By the standard construction, we obtain an automaton $\mathfrak{A}$ defining S and having $\sim$ as its equiresponse relation. The function g is constructed as follows. Let $[x]$ be an equivalence class under the relation $\sim$. If $[x] \subseteq S$, then, for all $y \in [x]$, $g(y, p) = f(y, p)$. If $[x]$ is not contained in S and for some w, $xw \in S$, then for all $y \in [x]$, $g(y, p)$ is specified so that

$$g(y, p) - g(x, p) = f(yw, p) - f(xw, p). \tag{11}$$

By virtue of (ii), the right-hand side of this expression is independent of w, so long as $xw \in S$; thus, in this case, (11) determines g up to an additive constant which is chosen arbitrarily. Finally, if $[x]$ is not contained in S and, further, there is no w such that $xw \in S$, then $g(x, p) = g(z, p)$, where z is the longest prefix of x such that, for some w, $zw \in S$.

To complete the proof of sufficiency, we shall specify the functions $k(p)$ and $h(\xi, q, a, p)$. Of course, $k(p) = g(e, p)$. Also, since the process II is required to be additive, the definition of h requires only the definition of an incremental cost function $I(q, a, p)$. This function must be related to the function g in the following way:

$$I(q, a, p) = g(ya, p) - g(y, p)$$

for all y such that $rp(y) = q$. Denote the equivalence class $\{y \mid \sim p(y) = q\}$ under the relation $\sim$ by $[x]$. Let us consider the case in which there exists a word w such that $xaw \in S$. Then, for any $y \in [x]$, let $I(q, a, p)$ be defined as follows:

$$\begin{aligned} I(q, a, p) &= g(ya, p) - g(y, p) \\ &= (g(xa, p) + f(yaw, p) - f(xaw, p)) \\ &\qquad - (g(x, p) + f(yaw, p) - f(xaw, p)) \\ &= g(xa, p) - g(x, p). \end{aligned}$$

Thus the specification of $I(q, a, p)$ is independent of the choice of $y \in [x]$. In the remaining case, where there is no w such that $xaw \in S$, it can be verified that $I(q, a, p)$ should be set equal to zero. This completes the proof of Theorem 3.

The following two corollaries, giving further information about the structures of additive representations, are easily established.

COROLLARY 2. *Let $\mathfrak{D} = (A, S, P, f)$ be a discrete decision process, and let $\sim$ be an equivalence relation over A^* satisfying the conditions of Theorem* 3.

Denote by $\hat{S}$ *the union of all equivalence classes* $[x]$ *such that, for some* w, $xw \in S$; *and let* $s(x, p)$ *be a real-valued function over* $\hat{S} \times P$. *Then there exists an additive sequential decision process* Π *which*

(i) *represents* $\mathfrak{D}$,

(ii) *has* $\sim$ *as its equiresponse relation,*

(iii) *has* $g(x, p) = s(x, p)$ *over* $\hat{S} \times P$

if and only if, whenever $x \sim y$ *and* $x \in \hat{S}$,

$$s(x, p) - s(y, p) = f(xw, p) - f(yw, p) \quad \text{for all} \quad p \in P,$$

where w *is an arbitrary word such that* $xw \in S$.

COROLLARY 3. *Let* $\mathfrak{D} = (A, S, P, f)$ *be a discrete decision process. Then* $\mathfrak{D}$ *is representable by an additive sequential decision process if and only if the following equivalence relation* $\sim$ *is of finite rank:* $x \sim y$ *if and only if*

(i) *for all* $w \in A^*$, $xw \in S$ *if and only if* $yw \in S$;

(ii) *for any two words* w *and* v *such that* $xw \in S$ *and* $xv \in S$, $f(xw, p) - f(yw, p) = f(xv, p) - f(yv, p)$ *for all* p.

Moreover, if $\sim$ *is of finite rank, it is the unique coarsest equivalence relation satisfying the conditions of Theorem* 3 (*i.e., any other such relation refines* $\sim$).

These two corollaries, taken together, establish that, whenever $\mathfrak{D}$ has an additive representation, a representation with a minimum number of states is determined uniquely except for certain arbitrary constants occurring in the function $g(x, p)$. It should be noted that a corresponding statement does not hold concerning the monotone representations of $\mathfrak{D}$, since there is not, in general, a unique coarsest relation satisfying the hypotheses of Theorem 2.

It is of interest to examine how Corollary 2 applies to the module placement problem discussed in §4. This problem was formulated as a discrete decision process $\mathfrak{D}^{(2)} = (A^{(2)}, S^{(2)}, P^{(2)}, f^{(2)})$, and an additive representation was given having the equiresponse relation $R_{S^{(2)}}$, and a cost function $g(x, p)$ specified in (9). There is a possibly more natural choice of a cost function. A simple lower bound on the cost $f^{(2)}$ of any policy beginning with the sequence $\sigma(1)\sigma(2) \cdots \sigma(r)$, where $\{\sigma(1), \sigma(2), \cdots, \sigma(r)\} = T$, is

$$\sum_{\substack{i,\, j=1 \\ i<j}}^{r} a_{\sigma(i),\,\sigma(j)}\,(j - i) + \sum_{\substack{i=1 \\ m \in \bar{T}}}^{r} a_{\sigma(i),\,m}\,(r + 1 - i).$$

This bound follows from the observation that, whatever the complete policy may be, object m is placed in a position with index at least $r + 1$, since the first r positions are already assigned. Thus, the distance between the position of object m and the position of object $\sigma(i)$ is at least $r + 1 - i$. Let us determine whether this lower bound $\hat{g}(x, p)$ can be used as the cost function in an additive representation of $\mathfrak{D}$ having the equiresponse relation

$R_{S^{(2)}}$. It follows directly from Corollary 2 that such a representation will exist if and only if, for each p, $\hat{g}(x, p) - g(x, p)$ is constant over each equivalence class $[x] \in \hat{S}^{(2)}$. If $x = \sigma(1)\sigma(2) \cdots \sigma(r)$, this difference is given by

$$(r + 1) \sum_{\substack{i=1 \\ m \in T}}^{r} a_{\sigma(i),m} .$$

Clearly the value of this expression depends only on the parameters and the set T, so that the condition is fulfilled, and the desired representation exists.

6. Some applications. In this section we discuss some typical examples of discrete, deterministic dynamic programming in order to demonstrate the applicability of the foregoing theoretical development.

6.1. 1-dimensional stock cutting. We consider a version of this problem in which it is desired to determine the minimum number of standard pieces of material of length L from which it is possible to cut a_1 pieces of length l_1, a_2 pieces of length l_2, $\cdots$, and a_n pieces of length l_n. To formulate this problem as a discrete decision process, we think of the required pieces as being cut successively, with a new standard piece introduced whenever the current standard piece does not leave enough material remaining for the next cut. The discrete decision process $\mathfrak{D} = (A, S, P, f)$ corresponding to this interpretation of the problem is as follows: $A = \{c_1, c_2, \cdots, c_n\}$, where the decision c_j has the interpretation "cut a piece of length l_j"; $S \subseteq A^*$ is the set of all sequences containing exactly a_j occurrences of c_j for $j = 1, 2, \cdots, n$; P is the set of all $(n + 1)$-tuples $(L, l_1, \cdots, l_n)$ of real numbers such that $l_j \leqq L$ for all j. Thus, the quantities n and a_j are fixed throughout the discussion; for, otherwise, S would not be independent of the data specification. The cost function $f(x, p)$ is defined in two stages. First, a function $s(x, p)$ is given recursively for all $(x, p) \in A^* \times P$ as follows:

$$s(e, p) = 0,$$

$$s(xc_j, p) = \begin{cases} s(x, p) + l_j & \text{if } s(x, p) + l_j \leqq L\left\lceil \dfrac{s(x, p)}{L} \right\rceil, \\ L\left\lceil \dfrac{s(x, p)}{L} \right\rceil + l_j & \text{otherwise.} \end{cases}$$

Here $\lceil \cdot \rceil$ denotes "least integer greater than or equal to." The function $f(x, p)$ is then defined to coincide with $s(x, p)$ over $S \times P$. Note that the second case in the definition of $s(xc_j, p)$ corresponds to the situation in which a new standard piece is required.

The right congruence R_S is easily seen to have the following form. One equivalence class consists of all sequences containing more than a_j occurrences of c_j for some j; otherwise, xR_Sy if and only if, for all j, the numbers of occurrences of c_j in x and y are equal. We shall establish that R_S satisfies the conditions of Theorem 2. It satisfies (i) by definition and (iii) vacuously, since no word in S is a prefix of another such word. To check that (ii) is satisfied, it is sufficient to notice that, if $s(x, p) \leqq s(y, p)$, then $f(xw, p) \leqq f(yw, p)$ for all w such that xw and yw are in S. Hence there exists a monotone sequential decision process $\Pi = (\mathfrak{A}, P, h, k)$ representing $\mathfrak{D}$ and having R_S as its equiresponse relation.

Setting $\sim$ equal to R_S in Corollary 1, we find that the function $s(x, p)$ satisfies the conditions of that corollary. Therefore, in constructing Π, we may set $g(x, p) = s(x, p)$. The function $k(p)$ must be set equal to $g(e, p)$, which is identically zero. The monotone function $h(\xi, q, a, p)$ must satisfy $h(g(x, p), q, a, p) = g(xa, p)$. From the recursive definition of $s(x, p)$ (which is identically equal to $g(x, p)$) it is evident that h may be specified as follows.

$$h(\xi, q, c_j, p) = \begin{cases} \xi + l_j & \text{if } \xi + l_j \leqq L\left\lceil \frac{\xi}{L} \right\rceil, \\ L\left\lceil \frac{\xi}{L} \right\rceil + l_j & \text{otherwise.} \end{cases}$$

It is apparent from the form of h that Π is not an additive process.

It is now possible to give the recurrence equations for the dynamic programming solution of the stock cutting problem. Let $(b_1, \cdots, b_n)$ denote the state corresponding to the equivalence class of R_S consisting of all sequences with exactly b_j occurrences of c_j for all j. Then, recalling that $G(q, p)$ denotes the minimum cost of reaching state q from the initial state, and applying Theorem 1 we obtain:

$$G((0, 0, \cdots, 0), p) = 0.$$

For $(b_1, \cdots, b_n) \neq (0, 0, \cdots, 0)$,

$$G((b_1, \cdots, b_n), p) = \min_{\{j \mid b_j \geqq 1\}} [h(G((b_1, \cdots, b_j - 1, \cdots, b_n), p), (b_1, \cdots, b_j - 1, \cdots, b_n), c_j, p)],$$

where the function h is as specified above.

In the case where $a_j = 1$ for all j, the stock cutting problem under consideration is isomorphic to a special case of an assembly-line-balancing problem previously considered by the authors [10], [11].

The most effective known method for treating large stock cutting prob-

lems is due to Gilmore and Gomory [8] and is based on linear programming. This method produces fractional solutions which must then be rounded to integer values. The inaccuracy due to rounding is inconsequential in large problems, but may be bothersome in small ones. Thus, a dynamic programming algorithm based on the above recurrence equations may prove useful for small problems.

The Gilmore-Gomory algorithm requires the repeated solution of a "knapsack problem". Problems of this type are particularly amenable to treatment by dynamic programming, as was first observed by Dantzig [6]. Let us apply our theory to the problem which Dantzig considered.

6.2. A knapsack problem. Suppose we are given n objects, where object j has weight a_j and value b_j. It is desired to pack a knapsack with a subset of these objects so as to maximize the value (minimize the negative of the value) without exceeding a given maximum weight W.

One formulation of this problem as a discrete decision process is specified by $\mathfrak{D} = (A, S, P, f)$ where: $A = \{1, 2, \cdots, n\}$, where j has the interpretation "place object j in the knapsack", S is the set of all sequences $i_1, i_2, \cdots, i_r$ such that $i_1 < i_2 < \cdots < i_r$ and $a_{i_1} + a_{i_2} + \cdots + a_{i_r} \leqq W$; P is the set of all n-tuples $(b_1, \cdots, b_n)$ of real numbers; the quantities n, W, and $a_1, \cdots, a_n$ are fixed, for otherwise S would not be independent of the parameters; and

$$f(i_1 i_2 \cdots i_r, p) = -(b_{i_1} + b_{i_2} + \cdots + b_{i_r}).$$

In seeking an equivalence relation satisfying the conditions of Theorem 3, we have usually begun by determining R_S. However, in this case it is difficult to specify R_S explicitly, and we choose to work instead with a more easily described right congruence $\sim$ refining R_S, given as follows:

$$x = i_1 \cdots i_r \sim y = j_1 \cdots j_s \quad \text{if and only if either} \quad x \notin S \text{ and } y \notin S$$
$$\text{or} \quad x \in S, \quad y \in S, \quad i_r = j_s,$$
$$\text{and } a_{i_1} + a_{i_2} + \cdots + a_{i_r} = a_{j_1} + a_{j_2} + \cdots + a_{j_s}.$$

The equivalence relation $\sim$ satisfies the conditions of Theorem 3 (as well as Theorem 2), so that an additive representation of $\mathfrak{D}$ may be expected. We may use the cost function $s(i_1 \cdots i_r, p) = -(b_{i_1} + b_{i_2} + \cdots + b_{i_r})$, since this function obviously satisfies the requirements of Corollary 2. It follows that $h(\xi, q, j, p) = \xi - b_j$. An additive sequential decision process Π representing $\mathfrak{D}$ is now determined. Let the state corresponding to the equivalence class of $\sim$ consisting of all words $i_1 \cdots i_r \in S$ such that $i_r = k$ and $a_{i_1} + \cdots + a_{i_r} = w$ be denoted (k, w). Note that not every ordered pair (k, w) with $k \leqq n$ and $w \leqq W$ corresponds to an equivalence class.

Then the dynamic programming recurrence equations are:

$$G((0, 0), p) = 0,$$
$$G((k, w), p) = \min (G((j, w - a_k), p) - b_k),$$

where the minimum is taken over all $j < k$ such that $(j, w - a_k)$ corresponds to an equivalence class.

A second, and perhaps more natural, formulation of the knapsack problem as a discrete decision process would be to define S as the set of all sequences $i_1 i_2 \cdots i_r$ such that $a_{i_1} + a_{i_2} + \cdots + a_{i_r} \leqq W$, thereby dropping the ascending index condition. However, this formulation would lead to a sequential decision process with many more states than the given one.

6.3. A routing problem.[3] A salesman based at city 1 must spend a day at each of the cities 2, 3, $\cdots$, n, and the overnight travel cost from city i to city j is c_{ij}. The salesman may make at most five stops during any given week, must be at home each week-end, and, once he returns during a given week, may not depart again until the next week. If the salesman's weekly base salary is C, find a schedule that minimizes the sum of his travel cost and salary for one complete tour of his territory. The following discrete decision process $\mathfrak{D} = (A, S, P, f)$ may be used to represent this problem: $A = \{1, 2, \cdots, n\}$, where j has the interpretation "go to the city j"; S is the set of all sequences beginning and ending with 1, such that each element of $\{2, 3, \cdots, n\}$ occurs exactly once, and not more than five elements of $\{2, 3, \cdots, n\}$ occur between successive occurrences of 1; P consists of all pairs $((c_{ij}), C)$, where (c_{ij}) is a positive real matrix and C is a positive constant; and

$$f(1i_2i_3 \cdots i_r1, p) = c_{1i_2} + c_{i_2i_3} + \cdots + c_{i_r1} + CM,$$

where $M + 1$ is the number of 1's in the sequence $1i_2i_3 \cdots i_r1$.

Note that x is a prefix of some word in S if and only if x begins with 1, has at most one occurrence of each index $j \neq 1$, and does not have more than five consecutive elements unequal to 1.

In this example xR_Sy if and only if

(1) neither x nor y is the prefix of a word in S, or

(2) x and y are both prefixes of words in S, and, in addition, (a) the sets of indices (cities) occurring in x and y are the same, and (b) if x and y are expressed uniquely as $x = a_x1b_x$ and $y = a_y1b_y$, where 1 does not occur in b_x or b_y, and t denotes the number of indices not in x (or y), then either $\log(b_x) = \log(b_y)$ or $\log(b_x) + t \leqq 5$ and $\log(b_y) + t \leqq 5$.

Any equivalence relation $\sim$ satisfying the conditions of Theorem 2 must,

[3] This problem is a variant of a truck dispatching problem considered in [5].

of course, refine R_S. Further, if $x \sim y$ and x and y have different final elements, then one can choose values for (c_{ij}) such that condition (ii) of Theorem 2 is violated. In particular, R_S violates condition (ii). Thus we are led to consider the following refinement $\perp$ of R_S : $x \perp y$ if and only if $x R_S y$ and x and y have the same final element. Then $\perp$ is the coarsest right congruence satisfying the conditions of Theorem 2 and, indeed, of Theorem 3. It follows that the given discrete decision process $\mathfrak{D}$ has an additive representation Π with $\perp$ as its equiresponse relation. In this representation we may set $h(\xi, q, j, p) = \xi + d_{ij}$, where every word in the equivalence class corresponding to q has i as its last element, and

$$d_{ij} = \begin{cases} c_{ij} + C & \text{if } j = 1, \\ c_{ij} & \text{if } j \neq 1. \end{cases}$$

Consider a particular equivalence class $[x]$ in $\perp$ consisting of a set of sequences containing exactly one occurrence of each element in a set $T \subseteq \{2, 3, \cdots, n\}$ and ending in l, where $l \in T \cup \{1\}$. The name of the corresponding state will be determined according to the following rules: if $l = 1$, the name is $(T, 1, 0)$; if $l \in \{2, 3, \cdots, n\}$ and, for every $y \in [x]$, $\log (b_y) = m$, where[4] $m + n - 1 - |T| > 5$, then the name is (T, l, m); if $l \in \{2, 3, \cdots, n\}$ and, for every $y \in [x]$, $\log (b_y) + n - 1 - |T| \leqq 5$, then the name is $(T, l, -)$.

The dynamic programming recurrence equations corresponding to Π are:

$$G((\varnothing, 1, 0), p) = 0,$$

$$G((T, 1, 0), p) = \min_{\{l,m \mid l \in T \cup \{1\}\}} (G((T, l, m), p) + d_{l1}),$$

$$G((T, l, m), p) = \min_{k \in T - l} (G((T - l, k, m - 1), p) + d_{kl}), \qquad m > 1,^5$$

$$G((T, l, 1), p) = G((T - l, 1, 0), p) + d_{1l},$$

$$G((T, l, -), p) = \min [\min_{k \in T - l} (G((T - l, k, -), p) + d_{kl}, \; G((T - l, 1, 0), p) + d_{1l})].$$

It may be noted that the classical traveling salesman problem, in which the salesman must visit all the cities before returning to city 1, leads to a similar set of recurrence equations (cf. [2], [10], [18]) involving only the states $(T, l, -)$.

So far our examples have been taken mainly from the field of operations

[4] $|T|$ denotes the number of elements of T.

[5] $T - l$ denotes the set obtained from T by deleting l.

research. Our methods are more widely applicable, as the following application to switching theory illustrates.

6.4. Finding a shortest reset sequence. Given a finite automaton $\bar{\mathfrak{A}} = \{\bar{A}, \bar{Q}, \bar{q}_0, \bar{\lambda}, \bar{F}\}$, find an input sequence x of shortest length such that $\bar{\lambda}(\bar{q}, x) = \bar{q}_0$ for all $\bar{q} \in \bar{Q}$. This problem has the following formulation as a discrete decision process $\mathfrak{D} = (A, S, P, f)$: $A = \bar{A}$, the input alphabet of $\mathfrak{A}$; $S = \{x \mid \bar{\lambda}(\bar{q}, x) = \bar{q}_0 \text{ for all } \bar{q} \in \bar{Q}\}$; no variable parameters occur, so P may be omitted from the discussion; and $f(x) = \log(x)$. Because of the simple nature of the cost function, conditions (ii) and (iii) of Theorem 2 are satisfied by any right congruence $\sim$. Condition (i) is satisfied if and only if $x \sim y$ implies for all w, $\bar{\lambda}(\bar{q}, xw) = \bar{q}_0$ for all $\bar{q}$ if and only if $\bar{\lambda}(\bar{q}, yw) = \bar{q}_0$ for all $\bar{q}$. Another way of writing this condition is: $\bar{\lambda}(\bar{\lambda}(\bar{q}, x), w) = \bar{q}_0$ for all $\bar{q}$ if and only if $\bar{\lambda}(\bar{\lambda}(\bar{q}, y), w) = \bar{q}_0$ for all q. Thus, in particular, the following right congruence $\sim$ satisfies the conditions of Theorem 2:

$$x \sim y \quad \text{if and only if} \quad \{\bar{\lambda}(\bar{q}, x) \mid \bar{q} \in \bar{Q}\} = \{\bar{\lambda}(\bar{q}, y) \mid \bar{q} \in \bar{Q}\}.$$

Setting $g(x) = \log(x)$, the following monotone sequential decision process $\Pi = (\mathfrak{A}, P, h, k)$ is obtained as a representation of $\mathfrak{D}$ (actually, we omit P, since it does not occur in $\mathfrak{D}$): $\mathfrak{A} = (A, Q, q_0, \lambda, F)$, where $A = \bar{A}$, Q is the set of all subsets of $\bar{Q}$, $q_0 = \bar{Q}$ (i.e., the subset consisting of all elements of $\bar{Q}$). For $S \subseteq \bar{Q}$ (equivalently, $S \in Q$), $\lambda(S, a) = \{\bar{\lambda}(\bar{q}, a) \mid \bar{q} \in S\}$ and $F = \{\{q_0\}\}$.

Finally, we set $k = 0$ and $h(\xi, q, a) = \xi + 1$, so that $g(x) = \log(x)$. The resulting dynamic programming recurrence equations are:

$$G(\bar{Q}) = 0,$$

$$G(S) = 1 + \min_{\{(T,a) \mid \lambda(T,a) = S\}} G(T) \quad \text{for} \quad S \neq \bar{Q}.$$

The cost of a shortest reset sequence is given by $G(\{q_0\})$.

In practice, most of the states of $\mathfrak{A}$ are inaccessible, and one would arrange the dynamic programming calculation so as to avoid consideration of inaccessible states.

7. Areas for further research. This paper provides a formal framework for the description and construction of dynamic programming algorithms for combinatorial optimization problems. Once such a problem has been formalized as a discrete decision process, the dynamic programming algorithms available for its solution are, in principle, completely determined. They are in one-to-one correspondence with the representations of the discrete decision process as a monotone sequential decision process. Two major difficulties arise, however. The first of these is that an informally

stated optimization problem can usually be cast as a discrete decision process in more than one way, as is illustrated by the module placement example (§4). In this connection, the problem of determining when two discrete decision processes correspond to the same "real world" problem should be investigated. The second difficulty is that no general procedure has yet been defined for finding an equivalence relation satisfying the conditions of Theorem 2 (or Theorem 3) and having a minimum number of equivalence classes. It is likely that this problem can be solved for the additive case provided a standard procedure is developed for specifying the cost function f. Until these difficulties are overcome, our approach will not be capable of systematically generating "best possible" dynamic programming algorithms, except in particular cases.

Another area deserving of attention is the study of the properties and methods of solution of the types of recurrence equations arising in discrete deterministic dynamic programming. These problems are well in hand in the additive case, where the theory developed for the shortest-route problem suffices [1], [15], as well as in the case where the number of policies is finite. In the latter case, none of the difficulties connected with "cycling" repeatedly through the states occur.

Finally, it remains to extend the entire theory to the stochastic case. It is likely that some generalization of Markovian decision processes [12] will play a role in this development.

REFERENCES

[1] R. E. Bellman, *On a routing problem*, Quart. Appl. Math., 16 (1958), pp. 87–90.

[2] ———, *Combinatorial processes and dynamic programming*, Proceedings of the Tenth Symposium in Applied Mathematics of the American Mathematical Society, 1960.

[3] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, 1962.

[4] D. Blackwell, *Discounted dynamic programming*, Ann. Math. Statist., 36 (1965), pp. 226–235.

[5] G. B. Dantzig, *The truck dispatching problem*, Management Sci., 6 (1959), pp. 80–91.

[6] ———, *Discrete variable extremum problems*, Operations Res., 5 (1957), pp. 266–277.

[7] E. V. Denardo, *Sequential decision processes*, Doctoral thesis, Northwestern University, Evanston, Illinois, 1965.

[8] P. C. Gilmore and R. E. Gomory, *A linear programming approach to the cutting stock problem*, Operations Res., 9 (1961), pp. 849–859.

[9] M. A. Harrison, *Introduction to Switching and Automata Theory*, McGraw-Hill, New York, 1965.

[10] M. Held and R. M. Karp, *A dynamic programming approach to sequencing problems*, this Journal, 10 (1962), pp. 196–210.

[11] M. Held, R. M. Karp and R. Shareshian, *Assembly-line balancing-dynamic*

programming with precedence constraints, Operations Res., 11 (1963), pp. 442–459.

[12] R. A. HOWARD, *Dynamic Programming and Markov Processes*, Technology Press, Cambridge, and John Wiley, New York, 1960.

[13] L. G. MITTEN, *Composition principles for synthesis of optimal multi-stage processes*, Operations Res., 12 (1964), pp. 610–619.

[14] G. L. NEMHAUSER, *Introduction to Dynamic Programming*, John Wiley, New York, 1966.

[15] M. POLLACK AND W. WIEBENSON, *Solutions of the shortest route problem — A review*, Operations Res., 8 (1960), pp. 224–230.

[16] M. RABIN AND D. SCOTT, *Finite automata and their decision problems*, IBM J. Res. Develop., 3 (1959), pp. 114–125.

[17] Y. A. SCHREIDER, *Automata and the problem of dynamic programming*, Problems of Cybernetics, 5 (1964), pp. 33–58.

[18] R. E. BELLMAN, *Dynamic programming treatment of the traveling salesman problem*, J. Assoc. Comput. Mach., 9 (1962), pp. 61–63.