MANAGEMENT SCIENCE
Vol. 12, No. 5, January, 1966
Printed in U.S.A.

DYNAMIC PROGRAMMING*†

RONALD A. HOWARD

Stanford University

Introduction

Dynamic programming is a mathematical technique for solving certain types of sequential decision problems. We characterize a sequential decision problem as a problem in which a sequence of decisions must be made with each decision affecting future decisions. We need to consider such problems because we rarely encounter an operational situation where the implications of any decision do not extend far into the future. For example, the best way to invest funds this year depends upon how the proceeds from this year's investments can be employed next year. The maintenance policy we should use for our machinery this year depends upon what we intend to do with this machinery in the future. The examples are as numerous as the fields of man's endeavor.

The Structure of Dynamic Programming

Dynamic programming is based on only a few concepts. Some it shares with other models; some are unique. The first concept that we must understand is that of a state variable. The state variables of a process are variables whose values completely specify the instantaneous situation of the process. The values of these variables tell us all we need to know about the system for the purpose of making decisions about it. The designation of system descriptors as state variables is quite arbitrary. For example, in the investment problem we might require as a state variable only the total amount of our present investment. Or we could define two state variables to describe income and investment growth. Or we might require one for investment in each industry, or even in each company. Although the number of state variables can theoretically be made as large as we please, the difficulty of solving the problems we face increases dramatically with the number of state variables involved. Thus it is to our advantage to minimize the number of state variables we use until any further simplification would destroy the utility of our model. In general, it is better to start simply and then complicate rather than to proceed in the reverse order. We usually speak of the values of all state variables as specifying the "state" of the system.

Having defined the state variables of the problem we introduce the concept of a decision as an opportunity to change these state variables, perhaps probabilistically. For example, the decision to sell a certain amount of one stock and buy another in the investment problem would lead to a change in state variables.

* Received March 1965.

† This is the fourth in a series of 12 expository papers commissioned jointly by the Office of Naval Research and the Army Research Office under contract numbers Nonr-4004(00) and DA 49-092-ARO-16, respectively.

However, because of the vagaries of the market, the net change in the state variables over some time period is subject to considerable uncertainty.

But why bother to make decisions affect the state? Because we can realize a profit or equivalently avoid a loss by having the state variables change in different ways. We imagine that each change of state has associated with it a reward, which may be negative. The rewards generated by each decision depend only on the starting and ending states for that decision and therefore may be added for a sequence of decisions. Our job is to make decisions that will make the total reward we shall achieve as high as possible.

Finally, we suppose that our ability to make decisions about the system occurs only at certain points of time which we shall call stages. At each stage we make a decision, change the state, and therefore make a reward. At the next stage we must make another decision using the values of the state variables that resulted from the preceding decision, and so on. Using these terms we can state our goal more precisely as a desire to maximize the expected total reward we shall receive when the number of stages available and the initial values of the state variables are fixed. In terms of the investment problem this might mean maximizing expected profit over a 10-year interval using a starting capital of $1,000.

We shall make all these ideas more precise in later sections, but first let us describe how dynamic programming solves such a problem and illustrate the procedure with some examples.

The Solution Concepts of Dynamic Programming

Perhaps the most interesting idea used in dynamic programming is that when we desire to maximize a function, the maximum value it can attain depends only on the constraints on the maximization and not on the procedure used to determine the maximum value. This is an idea best illustrated by examples. Suppose that we have available an artillery piece with a fixed muzzle velocity and that our job is to hurl the projectile as far as possible on level ground. The point is this: the maximum range we can achieve depends only on the muzzle velocity we are allowed. Those who studied basic physics know that if air resistance can be neglected, then the maximum range can be achieved by firing at an inclination of 45°. The actual range achieved will, of course, depend on the inclination angle used, but it will be a maximum only at the inclination 45°. Therefore if we picture ourselves as the sponsors of the gun-firing contest, we know that the only factor limiting the *maximum* range is the muzzle velocity we allow. The actual range achieved by any contestant will depend on this muzzle velocity and also on his intelligence in selecting the inclination angle. Thus we have illustrated that the *maximum* value of the function (range) depends only on the constraints (muzzle velocity) and not on the technique used to achieve it (inclination angle).

Perhaps another example will help. We all remember the problem of the farmer who wanted to enclose the maximum rectangular area with a fixed amount of fencing. We know that he will achieve the maximum area by using his fencing to surround a square plot. We might have solved the problem by trial and error, but more probably by calculus. Now suppose that we hold a fencing contest among farmers, supplying to each the same perimeter of fencing. We know that the maximum area anyone can enclose is determined only by the amount of fencing we provide and that it will be achieved only by a square design. Individual farmers may try a wide variety of rectangular solutions, with the one who comes closest to the square doing best. Here again we see that the *maximum* of the function (area enclosed) depends only on the constraints (fencing perimeter supplied) and not on the technique used to achieve it (shape of rectangle).

Since we have found that the best anyone can do in a decision problem depends only on the constraints, we find it worthwhile to postulate a portable genius. A portable genius is a person smart enough actually to achieve the maximum in a decision problem. The portable genius is a valuable resource—he knows that guns should be fired at a 45° inclination to achieve maximum range on level ground and that square plots have the maximum area of all rectangular plots with the same perimeter. We find it hard to hire people with this capability, but fortunately we shall find that we don't have to.

Once we have a portable genius we use him in solving a sequential decision problem by defining a function that is the total reward he could achieve if he were faced with a certain number of remaining stages and certain starting values of state variables. Since he can solve the problem, he can provide this total reward or value as we shall call it upon request. We are now ready to solve the sequential decision problem. In this problem we must first make one decision at the beginning of a long sequence of decisions. The problem is difficult because you and I as mere mortals seem to have no way of evaluating the influence of our present decision on the future. After all, how can we tell what is going to happen as a result of making this decision if we still don't know what we are going to do in the future? This is where we use our portable genius to break our multiple stage decision problem into single stage decision problems. We reason this way. In the present state we have available to us a number of different decisions that we could make. Each of these decisions would create some reward and place the system in some new state for the next stage. Now we call on our portable genius to tell us what we shall make in the future if we are now in this new state with one fewer stage remaining. He tells us, we add it to the reward from this stage, and thus obtain the total future profit from making this decision. Similarly we compute the total future profit from each other possible decision and then compare them. The decision with the highest total future profit is the one to choose.

Thus when we have the portable genius there is no difficulty in solving sequential decision problems where the profits are additive because he can tell us the implications for the future of any present action. If he tells us the value of being in each state with n stages remaining, then we can figure out what to do when we have $n + 1$ stages remaining. Thus if he tells us how we shall make out when there is only one stage remaining, we can compute the best decisions when there are two stages remaining. Now that we have evaluated what to do with two stages remaining we can compute for three stages remaining and so on. Therefore we need the portable genius only for the case when there is only one stage remaining. But we do not need him even here, because when there is only one stage remaining, there are no future decisions and the present decision can be based only on the profit that it will directly generate. Therefore, we do not need the portable genius at all, although he was a useful crutch in developing our thinking about sequential decision problems.

Therefore, sequential decision problems are solved by induction using the concept of values supplied by the portable genius, a genius who turns out never to be needed. We call this solution process dynamic programming, although as you can see, a better name might be recursive programming because of the form in which the solution is generated. Dynamic programming is quite different in form and concept from linear programming. Dynamic programming is conceptually more powerful and computationally less powerful than linear programming. An approximate analogy might be that dynamic programming is like calculus while linear programming is like solving sets of simultaneous linear equations. However, let us postpone further general comments on dynamic programming until we have achieved more feeling for the process through examples.

A Production Scheduling Example

Suppose that a company must produce a total quantity S in k monthly periods. We shall let n be the index for months remaining at any time and use p_n for the quantity that will be produced in the n^{th} month from the end of the k-month-long production period. Figure 1 illustrates a possible production plan. There are many production plans that meet the requirements

$$(1) \qquad \textstyle\sum_{n=1}^{k} p_n = S: \qquad p_n \geqq 0$$

Therefore, to make the problem definite, we assign a non-negative weighting factor w_n to the square of production in the n^{th} month from the end of the allowable time, and then require that the production plan minimize the sum of the weighted squares of production over the k-month period. The effect of the weighting factor would be to allow management to penalize production in some months more than others. We shall not discuss the merit of such a control procedure, but simply accept the result as an instructive example. Thus our problem is to

$$(2) \qquad \text{Min}_{p_n} \textstyle\sum_{n=1}^{k} w_n p_n^{2}$$

subject to the constraints in Equation 1.

We could solve this problem in many ways that range from trial and error to the calculus of variations. We choose to use dynamic programming and therefore proceed to cast the problem in dynamic programming terms. Our first task is to

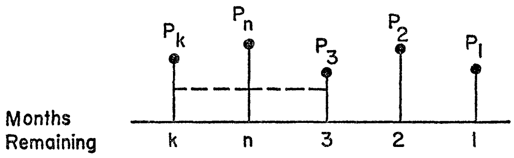


FIG. 1. A possible production plan

identify the state variables of the process. We see that if we enter the process in the middle of a production plan, we cannot affect the production levels that have already occurred—we can only influence the amounts to be made in each month in the future. What determines the weighted sum of squares of production we shall achieve in the future is the total amount s that remains to be made and the number of months n that are left in which to make the amount s. We can consider both n and s as the state variables of the process, or alternatively and for convenience we can call n the stage variable and s the state variable of the process. This terminology is reasonable because stages are the times at which decisions can be made and every remaining month is an opportunity to set a production level for that month.

Suppose that the weights w_n are fixed. If we present anyone with this problem and tell him n, the number of production periods he is allowed and s, the total amount he has to make, then in the light of our previous discussion we have already established the minimum weighted sum of squares that he will be able to obtain. Of course, he will not come up with a production plan to attain this minimum unless he is intelligent and diligent, but he can do no better than the minimum determined by n and s. We call this minimum $v(s \mid n)$, the value of being in state s with n stages remaining. It represents the minimum weighted sum of squares obtainable if the total amount to be made is s and n periods remain. Our portable genius is able to attain this minimum and to tell us its value when requested.

We also need a terminology for the decision in order to specify the production plan that will be our answer. We let $p(s \mid n)$ be the amount to be produced in the n^{th} period from the end if a quantity s remains to be made in n monthly periods. If we knew this function, we would find it very easy to construct a production plan.

Let us now consider the situation where we have an amount s to be made in n months. If we make an amount p_n in the present month, then we shall have to make a total quantity $s - p_n$ in the remaining $n - 1$ months. The contribution to the weighted sum of squares of making p_n in the present month is $w_n p_n^2$. The minimum weighted sum of squares that can be achieved in the remaining $n - 1$ months starting with a quantity $s - p_n$ to be produced is reported by our portable genius to be $v(s - p_n \mid n - 1)$. Therefore the total weighted sum of squares we shall achieve by making p_n now and doing the best anyone can do in the remaining months is $w_n p_n^2 + v(s - p_n \mid n - 1)$. Of course, we want to choose p_n so as to make this total weighted sum of squares as small as possible. If we do, then we can say that this quantity is $v(s \mid n)$, the best that can be done with a total quantity s to produce in n months. We therefore have the equation

$$(3)\quad \begin{aligned} &v(s \mid n) \\ &\quad = \mathrm{Min}_{0 \leqq p_n \leqq s} \{w_n p_n^2 + v(s - p_n \mid n - 1)\}, \qquad s \geqq 0,\ n = 2, 3, 4, \cdots. \end{aligned}$$

Note that we restrict the quantity p_n that can be made in the present month to be non-negative and to be no more than s, the total quantity to be made throughout the n month period. Of course the value of p_n that minimizes the sum in Equation 3 is $p(s \mid n)$, the amount that should be made in the n^{th} month from the end of an n-month period in which a total quantity s is to be produced so as to minimize the weighted sum of squares of production. Notice that the sum is composed of two parts, the immediate effect of the decision to make p_n, $w_n p_n^2$, and the long run effect of the decision, $v(s - p_n \mid n - 1)$. This structure is typical of the dynamic programming approach: it provides the correct balance of short and long run consequences.

Well, we have solved the problem if we have a portable genius, but we don't have him. What do we do? We say that when only one month remains we don't need him and that Equation 3 then shows us how to dispense with him when more than one month remains. Let us consider the situation when there is only one month in which to make a total quantity s. Then there is not much choice—the total quantity s must be produced in this month and the weighted sum of squares incurred will be $w_1 s^2$,

$$\begin{aligned} p(s \mid 1) &= s \\ v(s \mid 1) &= w_1 s^2. \end{aligned} \tag{4}$$

Thus the problem becomes trivial when only one month remains.

But now that we know what to do with one month remaining we can figure out what to do with two months remaining by writing Equation 3 with $n = 2$,

$$v(s \mid 2) = \text{Min}_{0 \leq p_2 \leq s} \{w_2 p_2^2 + v(s - p_2 \mid 1)\}. \tag{5}$$

Since we already know the function $v(\cdot \mid 1)$ from Equation 4, the minimization is clearly identified. We have many techniques that can be applied to perform this type of minimization. The easiest one to use in this case is differentiation of the expression and setting the result equal to zero. When we differentiate the quantity in braces with respect to p_2 and set it equal to zero, we obtain

$$\partial\{\quad\}/\partial p_2 = 2w_2 p_2 - v'(s - p_2 \mid 1) = 0. \tag{6}$$

From Equation 4 we observe

$$v'(s \mid 1) = 2w_1 s \tag{7}$$

so that we can write Equation 6 as

$$2w_2 p_2 - 2w_1(s - p_2) = 0 \tag{8}$$

or

$$p_2 = p(s \mid 2) = w_1 s/(w_1 + w_2). \tag{9}$$

Thus if we have two months in which to make a total quantity s we should make a fraction $w_1/(w_1 + w_2)$ of the quantity in the present month to assure that the weighted sum of squares will be a minimum.

We find $v(s \mid 2)$, the minimum weighted sum of squares that we can obtain in producing a total quantity s in a two month period by evaluating the quantity in braces in Equation 5 at the minimizing value of p_2 given by Equation 9. We obtain

$$
(10)\quad
\begin{aligned}
v(s \mid 2) &= \{w_2 p_2^2 + v_{\cancel{1}}(s - p_2)\}_{p_2 = w_1 s/(w_1 + w_2)} \\
&= w_2(w_1/(w_1 + w_2))^2 s^2 + v_{\cancel{1}}(w_2 s/(w_1 + w_2)) \\
&= w_2 w_1^2 s^2/(w_1 + w_2)^2 + w_1 w_2^2 s^2/(w_1 + w_2)^2 \\
&= w_1 w_2 s^2/(w_1 + w_2).
\end{aligned}
$$

This is, of course, just the weighted sum of squares that will result from making $w_1 s/(w_1 + w_2)$ in the present month and $w_2 s/(w_1 + w_2)$ in the last month.

Equation 9 yields the policy to follow at the second stage, the amount that should be produced at present to insure that the decision will be the best possible as far as future decisions are concerned. Equation 10 provides the value, the expected profit that will result by taking the optimum present decision and also following an optimum course in the future.

Now that we have evaluated what to do when 1 or 2 months remain we can continue to the case where three months remain. Then Equation 3 takes the form

$$(11)\quad v(s \mid 3) = \mathrm{Min}_{0 \leq p_3 \leq s}\{w_3 p_3^2 + v(s - p_3 \mid 2)\}.$$

We again perform the minimization by differentiation and write

$$(12)\quad \partial\{\ \}/\partial p_3 = 2w_3 p_3 - v'(s - p_3 \mid 2) = 0.$$

Equation 10 shows that the derivative of the function $v(s \mid 2)$ with respect to its argument is

$$(13)\quad v'(s \mid 2) = 2w_1 w_2 s/(w_1 + w_2).$$

When we substitute this result into Equation 12, we find

$$
(14)\quad
\begin{aligned}
2w_3 p_3 - 2w_1 w_2(s - p_3)/(w_1 + w_2) &= 0, \\
p_3(w_1 w_2 + w_1 w_3 + w_2 w_3) &= w_1 w_2 s
\end{aligned}
$$

or

$$(15)\quad p_3 = p(s \mid 3) = w_1 w_2 s/(w_1 w_2 + w_1 w_3 + w_2 w_3).$$

The amount that should be made at present if three months remain in which to make a total quantity s is a fraction of s equal to the product of the weights in the last two months divided by the sum of the products of all possible pairs of weights for the three months.

If we substitute this value for p_3 into Equation 11, we obtain $v(s \mid 3)$, the minimum weighted sum of squares obtainable throughout the three month period,

$$
(16)\quad
\begin{aligned}
v(s \mid 3) &= \{w_3 p_3^2 + v(s - p_3 \mid 2)\}_{p_3 = w_1 w_2 s/(w_1 w_2 + w_1 w_3 + w_2 w_3)} \\
&= \{w_3 p_3^2 + w_1 w_2 (s - p_3)^2/(w_1 + w_2)\}_{p_3 = w_1 w_2 s/(w_1 w_2 + w_1 w_3 + w_2 w_3)} \\
&= w_1 w_2 w_3 s^2/(w_1 w_2 + w_1 w_3 + w_2 w_3).
\end{aligned}
$$

Thus Equation 15 provides the policy or what to do and Equation 16 provides the value or how much we shall profit by doing it for the case when three months remain.

We could continue to use the recursive Equation 3 to solve the problem for higher and higher numbers of months or stages remaining. However, what we have already done is sufficient to see the form of the solution. For the policy we observe that

$$(17) \qquad p(s \mid n) = \frac{1/w_n}{\sum_{i=1}^{n} 1/w_i} s, \qquad n = 1\ 2, 3, \cdots$$

is consistent with the results of Equations 4, 9, and 15. It is clear that the reciprocals of the weights appear more naturally in the problem than do the weights themselves. Similarly, for the value we see that

$$(18) \qquad v(s \mid n) = s^2 / \textstyle\sum_{i=1}^{n} 1/w_i$$

ir a general form for the results of Equations 4, 10, and 16. Furthermore, if we substitute the policy and value results from Equations 17 and 18 directly into Equation 3, we find that Equation 3 is satisfied. Therefore Equations 17 and 18 constitute a complete solution to the problem.

A word is in order about the form in which the solution has been generated. Although we seek a production plan for a specific total quantity S to be made in a time period specified to be k months long, we have in fact solved a more general problem. The more general problem is that of finding how much to make in the present month for any amount to be produced s and months remaining n. We can not only solve the specific problem proposed, but also the whole class of probems. This generality provided and required by the dynamic programming procedure is called imbedding the problem in a larger class of problems. As you might expect, imbedding is a very good idea when you would like to have a general solution. However, it may also lead to higher computational requirements than would be needed if the specific problem posed were solved directly. The point is that in dynamic programming you always get more than you bargained for, even if you don't want it.

Note that when we finally get around to solving a specific problem, we must interpret our general solution as a rule for generating and evaluating optimum production plans rather than as a production plan itself. The policy tells us how much to make in the present month if we know the total amount to be made and the number of months remaining. Once we take this action, we then proceed to the next decision point with a lesser quantity to be made and one less month remaining. The policy for this stage again tells what to make in the present month, and we continue to generate the production plan in this way. However, at any stage and state the value function tells us the minimum sum of squares we shall be able to obtain into the future, even though at this point we have not yet found the explicit production plan that will be followed in the future. In other words, the value function tells our farmer contestant the maximum rectangular area he will be able to enclose even before it tells him the dimensions to use. Thus the policy and value functions provide powerful, if somewhat unusual, aid in solving decision problems.

Perhaps the structure of the solution appears most clearly in the case when all weights are equal, and for simplicity, let us say equal to unity. Then the policy of Equation 17 becomes

$$p(s \mid n) = s/n, \qquad n = 1, 2, 3, \cdots . \tag{19}$$

This equation states that when n months remain in which to make a total quantity s, an amount s/n should be made in the present month. We know from the symmetry of the problem that when all weights are equal, equal amounts should be produced in each of the remaining months. Equation 19 presents this result in implicit rather than explicit form. For example, suppose that three months remain to make a total quantity S. Then according to Equation 19, $\frac{1}{3}S$ should be made in the present month. Next month a quantity $\frac{2}{3}S$ will remain; Equation 19 shows that one half of it or $\frac{1}{3}S$ should be produced next month. That will leave $\frac{1}{3}S$ to be produced during the last month, and Equation 19 states, of course, that all of it must be produced in that month. The net result is that we have produced $\frac{1}{3}S$ in each of the three remaining months by applying the result of Equation 19 recursively.

Equation 18 shows that the value function for the case of equal unity weights is

$$v(s \mid n) = s^2/n, \qquad n = 1, 2, 3, \cdots . \tag{20}$$

For the production schedule we have just developed the weighted sum of squares is

$$(\tfrac{1}{3}S)^2 + (\tfrac{1}{3}S)^2 + (\tfrac{1}{3}S)^2 = \tfrac{1}{3}S^2, \tag{21}$$

which is the result of Equation 20 when a total quantity S must be made in three months. The interpretation of a policy as a decision rule rather than as a specific plan and the interpretation of the value function as an overall measure of policy rather than of any part of it are fundamental to the dynamic programming approach.

Numerical Results

Suppose that we want to minimize the weighted sum of squares of production of a three month period in which we must make 9 units. Thus $k = 3$, $S = 9$. The weighting factors for the three months are $w_1 = 2$, $w_2 = 3$, $w_3 = 6$ and are shown in Figure 2. We begin by deciding how much to make in month 3 (from

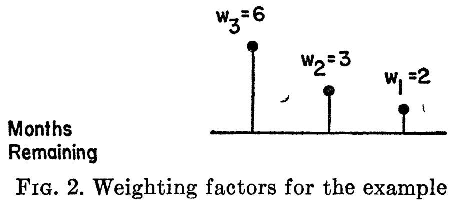


FIG. 2. Weighting factors for the example

the end). From Equation 17,

$$(22) \qquad p(9 \mid 3) = \frac{1/w_3}{1/w_1 + 1/w_2 + 1/w_3}(9) = \frac{\frac{1}{6}}{\frac{1}{2} + \frac{1}{3} + \frac{1}{6}}(9) = 1.5.$$

We find that we should make 1.5 units in the present month when three months remain. Manufacturing 1.5 units might be a problem if our products were automobiles, but we shall assume that it is a bulk product for the moment and return to the quantization problem later.

From Equation 18 we compute the value of being in state 9 with three stages left,

$$(23) \qquad v(9 \mid 3) = \frac{1}{1/w_1 + 1/w_2 + 1/w_3}(9)^2 = \frac{1}{\frac{1}{2} + \frac{1}{3} + \frac{1}{6}}(81) = 81.$$

Thus the minimum weighted sum of squares we shall be able to obtain is 81. We have not yet specified what our production plan will be (except for the first month and we have not used the result of that calculation) and still we can predict how well we shall do by following it.

Now we come to two months remaining with a total quantity $9 - 1.5 = 7.5$ yet to be produced. The amount of production for this middle month we compute from Equation 17 as

$$(24) \qquad p(7.5 \mid 3) = \frac{1/w_2}{1/w_1 + 1/w_2}(7.5) = \frac{\frac{1}{3}}{\frac{1}{2} + \frac{1}{3}}(7.5) = 3.$$

From Equation 18 we compute the minimum weighted sum of squares for the last two months,

$$(25) \qquad v(7.5 \mid 3) = \frac{1}{1/w_1 + 1/w_2}(7.5)^2 = \frac{1}{\frac{1}{2} + \frac{1}{3}}(56.25) = 67.5.$$

Notice that since our production of 1.5 units in the third month from the end contributed $w_3 p_3^2 = 6(1.5)^2 = 13.5$ to the weighted sum of squares, the predictions of Equations 23 and 25 are consistent.

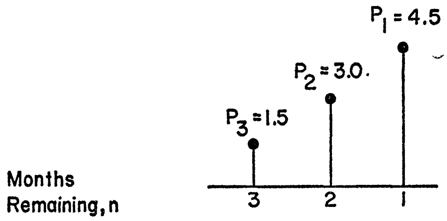
In the last month remaining we have left to produce $9 - 1.5 - 3 = 4.5$ units, and of course Equation 17 states that they all must be produced in this month,

$$(26) \qquad p(4.5 \mid 1) = 4.5.$$

Equation 18 then shows that the contribution to the sum of squares of this last month is

$$(27) \qquad v(4.5 \mid 1) = \frac{1}{1/w_1}(4.5)^2 = 2(20.25) = 40.5.$$

The complete production plan along with the weighted squares of production for the plan appears in Figure 3. We observe that the predictions of Equations 23, 25, and 27 about the sum of squares to be obtained in the future of the plan at different stages are substantiated by these results. The sum of squares for the



P_n	1.5	3.0	4.5	
P_n^2	2.25	9.0	20.25	
$w_n p_n^2$	13.50	27.0	40.50	$\sum_{n=1}^{3} w_n p_n^2 = 81$

FIG. 3. The best production plan

entire plan is 81 as predicted initially by $v(9 \mid 3)$. No other production plan can achieve so low a weighted sum of squares.

The Variational Approach

We observe that the best production plan has the property that the product of the production and weighting factor in any month is the constant 9. We can explain this behavior by solving this same problem using a variational approach. Thus to perform the minimization of Equation 2 subject to the constraints of Equation 1 we would first form the quantity

$$\sum_{n=1}^{k} w_n p_n^2 + \lambda\left(\sum_{n=1}^{k} p_n - S\right) \tag{28}$$

and then differentiate it both with respect to p_n and with respect to λ. We obtain for these differentiations

$$2\,w_n p_n + \lambda = 0 \tag{29}$$

and

$$\sum_{n=1}^{n} p_n = S. \tag{30}$$

Then

$$p_n = -\lambda/2w_n \tag{31}$$

and we use Equation 30 to find λ,

$$\sum_{n=1}^{k} p_n = S = -(\lambda/2)\sum_{n=1}^{k} 1/w_n \tag{32}$$

or

$$\lambda = -2S\Big/\sum_{n=1}^{k} 1/w_n\,. \tag{33}$$

Finally we substitute this result into Equation 31 to produce the solution

$$p_n = \frac{1/w_n}{\sum_{n=1}^{k} 1/w_n}\,S, \qquad n = 1, 2, \cdots, k. \tag{34}$$

Happily the production quantities turn out to be positive so we do not have to worry about the constraint that they be positive.

Equation 31 shows immediately that the product of production quantity and weighting factor in any month will be a constant. Equation 33 shows further that this constant is just the total quantity to be produced divided by the sum of the reciprocals of the weighting factors for the months involved, and we have therefore checked our numerical example.

However, the important thing to note in the variational solution is that Equation 34 describes the production quantity for each month rather than a rule for constructing the production quantity when the amount yet to be made is known. The solution is explicit rather than in the recursive form of Equation 17. Since both expressions look virtually identical, they differ only in interpretation, but the difference is critical. If we evaluate the weighted sum of squares of production for this production plan, we find

$$\sum_{n=1}^{k} w_n\, p_n^2 = \sum_{n=1}^{k} w_n \frac{(1/w_n)^2}{(\sum_{n=1}^{k} 1/w_n)^2} S^2 = \frac{1}{\sum_{n=1}^{k} 1/w_n} S^2, \tag{35}$$

in agreement with Equation 18 when S must be produced in k months.

It is clear that solving the production problem by variational methods is simpler than solving it by dynamic programming. We chose the less efficient route initially because it is a good way to illustrate the dynamic programming process. Yet if variational methods are better, why do we need dynamic programming? The answer is that dynamic programming provides a convenient approach to problems that are difficult to treat by variational methods. For example, the requirement that the production quantities must be positive is difficult to incorporate in the variational approach. We were fortunate that this constraint was not violated in the present problem. But what if it had been? The dynamic programming approach can easily handle this type of difficulty. At most it would mean that we could not use the differentiation method for finding the minimum of the quantities in braces in Equations 5 and 11; however, other methods are available. Similarly, we can use dynamic programming to treat the case where there are limits on the production in each month. But perhaps the most dramatic illustration of the advantage of dynamic programming arises when we require that the amount of production in each month be an integer.

The Production Problem with Integer Constraints

Let us consider the same example of producing 9 units in three months so as to minimize the weighted sum of squares of production using the weights of Figure 2. Only now we shall require that the number of units made during each month be integral. The only change that we have to make in our previous analysis is that the values of p_n considered in carrying out the minimization in Equation 3 must be limited to the integer values 0 through s; that is,

$$v(s \mid n) = \operatorname{Min}_{p_n=0,1,2,\cdots,s}\{w_n p_n^2 + v(s - p_n \mid n - 1)\}, \qquad s = 0, 1, 2, \cdots, n = 2, 3, 4, \cdots . \tag{36}$$

TABLE 1

Computation of Integral Production Problem at First Stage

s	$p(s) \mid 1) = s$	$v(s) \mid 1) = w_1s^2 = 2s$
0	0	0
1	1	2
2	2	8
3	3	18
4	4	32
5	5	50
6	6	72
7	7	98
8	8	128
9	9	162
10	10	200

Since the quantity to be made must always be specified as an integer, Equation 4 still applies,

$$\begin{aligned} p(s \mid 1) &= s, \\ v(s \mid 1) &= w_1 s^2. \end{aligned} \tag{37}$$

Consequently, Equations 36 and 37 constitute a complete formal statement of the problem of finding the best integral production schedule.

We begin writing $p(s \mid 1)$ and $v(s \mid 1)$ in Table 1 for values of s ranging from 0 through 10. We choose 10 simply to illustrate that we are not solving a particular problem, but rather a class of problems.

We continue now to the second stage and write Equation 36,

$$\begin{aligned} v(s \mid 2) &= \text{Min}_{\,p_2=0,1,2,\cdots,s}\{w_2 p_2^2 + v(s - p_2 \mid 1)\} \qquad s = 0, 1, 2, \cdots \\ &= \text{Min}_{\,p_2=0,1,2,\cdots,s}\{3\, p_2^2 + v(s - p_2 \mid 1)\}. \end{aligned} \tag{38}$$

For $s = 0$,

$$v(0 \mid 2) = \text{Min}_{\,p_2=0}\,\{3\, p_2^2 + v(-p_2 \mid 1)\} = v(0 \mid 1), \tag{39}$$

an obvious result. For $s = 1$,

$$\begin{aligned} v(1 \mid 2) &= \text{Min}_{\,p_2=0,1}\,\{3\, p_2^2 + v(1 - p_2 \mid 1)\} \\ &= \text{Min}\begin{cases} p_2 = 1: & 3 + v(0 \mid 1) = 3 + 0 = 3 \\ p_2 = 0: & 0 + v(1 \mid 1) = 0 + 2 = 2 \end{cases} \\ &= 2 \text{ at } p_2 = 0. \end{aligned} \tag{40}$$

Therefore when one unit must be made in two months, it should not be made in the present month, but made in the last month. The weighted sum of squares so incurred will be 2, For $s = 2$,

$$v(2 \mid 2) = \operatorname{Min}_{p_2 = 0,1,2} \{3p_2^2 + v(2 - p_2 \mid 1)\}$$

$$(41) \qquad = \operatorname{Min}\begin{cases} p_2 = 2: & 12 + v(0 \mid 1) = 12 + 0 = 12 \\ p_2 = 1: & 3 + v(1 \mid 1) = 3 + 2 = 5 \\ p_2 = 0: & 0 + v(2 \mid 1) = 0 + 8 = 8 \end{cases}$$

$$= 5 \text{ at } p_2 = 1.$$

Therefore when there are two units to be made in two months, one should be made in each month; a weighted sum of squares of 5 will be achieved.

We could continue in this way to compute all values of $v(s \mid 2)$ from $v(s \mid 1)$, but let us examine the general procedure we have been following. Suppose $s = 3$. Then we are constructing a column $3p_2^2$ for $p_2 = 3, 2, 1, 0$ and adding to each entry the value of $v(4 - p_2 \mid 1)$ from Table 1, thus,

(42)

p_2	$3p_2^2$		$v(4 - p_2 \mid 1)$		
3	27	+	0	=	27
2	12	+	2	=	14
1	3	+	8	=	11
0	0	+	18	=	18.

The smallest value of the sum is 11, occurring when $p_2 = 1$. Therefore if 3 units must be made in two months, one should be made in the present month and two in the last month. The total weighted sum of squares will then be 11.

Thus a simple way to construct $v(s \mid 2)$ would be to construct a paper strip with a column containing entries p_2 and $3p_2^2$. This strip could be moved beside the column $v(s \mid 1)$ from Table 1 and the value of $3p_2^2 + v(s - p_2 \mid 1)$ could be computed mentally through row s of Table 1. The smallest value of this quantity would be $v(s \mid 2)$; the value of p_2 at which it occurred would be $p(s \mid 2)$. Then the strip would be moved down one space and the computation would be repeated for the next higher value of s, and so on. By proceeding in this way with a paper strip the calculation of $v(s \mid 2)$ from $v(s \mid 1)$ becomes straightforward. When we continue on to calculate $v(s \mid 3)$ from $v(s \mid 2)$, we use the same procedure, but use a paper strip containing the values of p_3 and $6p_3^2$ since $w_3 = 6$. The values on this strip are added mentally to the values of $v(s - p_2 \mid 2)$ already obtained to perform the optimization at the third stage. Since we can continue this process indefinitely, we can find $v(s \mid n)$ for any value of n we like.

Table 2 shows the result of following this computational procedure for the first three stages. The table provides a solution for any number of months remaining through 3 and any quantity to be produced through 10. To see how to use this table, suppose we determine the production plans corresponding to the requirement of producing 9 units in three months. From the table we observe $p(9 \mid 3) = 1, 2$. That is, when we have three months remaining and 9 units to make, we have a choice of making either 1 or 2 in the present month. The choice arises because we find in determining the minimum value of $v(9 \mid 3)$ in the computational procedure that two different values of p_3, 1 and 2, produce the same minimum value of 83. Suppose that we decide to make 1 unit in month 3. Then

TABLE 2

Result of Three Stages for Integral Production Problem

s	$p(s \mid 1)$	$v(s \mid 1)$	$p(s \mid 2)$	$v(s \mid 2)$	$p(s \mid 3)$	$v(s \mid 3)$
0	0	0	0	0	0	0
1	1	2	0	2	0	2
2	2	8	1	5	0	5
3	3	18	1	11	0, 1	11
4	4	32	2	20	1	17
5	5	50	2	30	1	26
6	6	72	2	44	1	36
7	7	98	3	59	1	50
8	8	128	3	77	1	65
9	9	162	4	98	1, 2	83
10	10	200	4	120	2	101

8 units must be made in the last two months. Since $p(8 \mid 2) = 3$, we find that we must make 3 units in month 2. That leaves $9 - 1 - 3 = 5$ units to be made in month 1, and, of course, $p(5 \mid 1) = 5$.

If on the other hand we choose to make 2 units in month 3, then $9 - 2 = 7$ units remain to be made in the last two months. Since $p(7 \mid 2) = 3$, again we make 3 units in month 2. That leaves $9 - 2 - 3 = 4$ units to be made in month 1, and $p(4 \mid 1) = 4$. Therefore we have our choice of two production plans:

$$\begin{array}{llll} \text{Plan 1} & p(9 \mid 3) = 1 & p(8 \mid 2) = 3 & p(5 \mid 1) = 5 \\ \text{Plan 2} & p(9 \mid 3) = 2 & p(7 \mid 2) = 3 & p(4 \mid 1) = 4 \end{array} \tag{43}$$

We can make the 9 units in the order 1, 3, 5 or in the order 2, 3, 4. The monthly contributions to the weighted sum of squares for each plan are:

$$\begin{array}{llcccc} & & n=3 & n=2 & n=1 & \text{Total} \\ \text{Plan 1} & w_n p_n^2 & 6 & 27 & 50 & 83 \\ \text{Plan 2} & w_n p_n^2 & 24 & 27 & 32 & 83 \end{array} \tag{44}$$

We see that both plans produce a weighted sum of squares of 83. We observe that $v(8 \mid 2) = 77$ gives the correct prediction of the weighted sum of squares in the last two months for plan 1 and that $v(7 \mid 2) = 59$ gives the corresponding correct prediction for plan 2.

Thus we have seen that the requirement that the amount of production in each month be integral has caused the minimum weighted sum of squares to increase by two units from the value of 81 we observed in Equation 23 for the case where this constraint was not imposed. This points up the general result that adding a constraint in a problem can never help and may hinder the attainment of the maximum or minimum value obtainable without the constraint. The effect of adding constraints has philosophical as well as mathematical implications.

Finally, we should note that the value function must be unique, but the policy function need not be. The value function is unique because it represents the results obtainable from our portable genius—the best anyone could do. Clearly the best cannot have two different values. However, the policy need not be unique. As in this problem, there may be two or more different sets of decisions that lead to the same optimum solution of the decision problem. Many roads of equal length may lead to Rome. We should regard this possibility with gratitude because it allows us in some situations a set of policies that are equally good from a formal point of view, but within which we may be able to exercise a preference on some second order basis, like ease of explanation. Of course, we could also include more and more of these second order considerations into the formal statement of the problem and ultimately make the optimum policy unique. Rarely, however, is such an approach necessary or desirable.

An Action Timing Problem

As a further example of dynamic programming we consider what we shall call an action-timing problem. This is a problem in which a certain action may be taken only once and the question is when it should be taken. The problem we shall solve could serve as a useful model for a wide variety of situations. For example, suppose that we are riding in an automobile and will soon enter a turnpike. We are low on gasoline, and decide to fill up. However, we know the gasoline on the turnpike is expensive and that the stations before the turnpike vary in their gasoline prices. The problem is if we know the number of stations remaining and observe the price at the present station, should we or should we not fill up here? The same problem could arise in hiring if we were allowed to interview 10 applicants for a single vacancy but had to reach an irrevokable decision not to hire each one after he was interviewed before we could interview the next applicant. Or we might encounter the problem in a commodity market where we had to make a purchase before a certain deadline and we wanted to know whether today's price is sufficiently low to make the purchase. Even in amateur photography can we find an application. Suppose that we have only one frame of film left and no possibility of obtaining any more. How do we determine whether a given scene is worthy of a shot when we have only expectations about the scenes we have remaining? Finally, the problem of getting married can be posed in these terms, but perhaps it is not wise to pursue that example further.

We shall develop a model general enough to allow considering all these possibilities. A known number of opportunities for action will be presented to us. At any of them we can take action and thereby terminate the problem. We can observe the utility of each opportunity before deciding whether or not to act. However, we do not know the utilities of future opportunities. We assume that the utility of each opportunity is measured on a scale ranging from 0 to 1 with 0 being the lowest possible utility and 1 the highest. Thus the utility of one corresponds to the cheapest possible gasoline, the most capable employee, the best possible commodity bargain, the unforgettable picture, and the ideal wife. We assume further that the utilities u presented on successive opportunities are selected independently from the same probability distribution. Our initial choice for this distribution is the uniform distribution of Figure 4. All values of utility between 0 and 1 are equally likely to be presented at each opportunity.

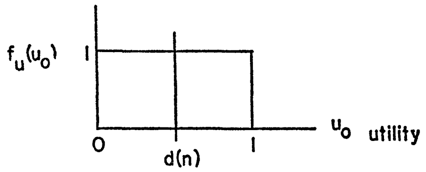


FIG. 4. The density function for the utility presented at each opportunity

We shall let n represent the number of opportunities remaining at any time; in our previous terminology n is a stage variable. The essential problem is this: if we have arrived at a point where we have n opportunities remaining and have not yet acted, should we accept the present utility and act now? What we would really like to have is a decision criterion $d(n)$ for accepting the utility u offered with n opportunities remaining. That is, we would like to know a number $d(n)$ such that if u is less than $d(n)$ we shall not act now, while if u equals or exceeds $d(n)$ we shall act now. If we denote acting by A and not acting by A', then at any n

$$\begin{aligned} &u < d(n) \quad \text{implies} \quad A' \\ &u \geqq d(n) \quad \text{implies} \quad A. \end{aligned} \tag{45}$$

To calculate $d(n)$, we begin by noting that the decision maker must be in one of two states at any time. If we let i be the index of his state, then we can assign $i = 1$ to the situation where he has not already taken the action and $i = 2$ to the situation where he has already acted. We define $v_i(n)$ as the value or expected utility of being in state i when n opportunities remain. It is often convenient as it is here to specify the state of the system by a subscript. Since there is no value of being in state 2 beyond that received when the action was taken, we have

$$v_2(n) = v_2(n-1) = \cdots = v_2(0) = 0. \tag{46}$$

However, there is an expected utility to be derived from being in state 1 when n opportunities remain. If the decision maker sets the criterion level $d(n)$ at some value, then he will take action if u is at least as large as $d(n)$ and otherwise not act. Since the utility u that will be presented is a random variable, the probability that he will act when u is revealed is the probability that u will equal or exceed $d(n)$. If he does act, then he will obtain the expected utility of a draw conditional on the draw's not being less than $d(n)$. He will also receive the value of being in state 2, but as we have said this position has no value in itself. The probability that he will not take action is 1 minus the probability that he will take action. In this case he finds himself once more in state 1, but this time with one fewer opportunity remaining. The value of this situation is $v_1(n-1)$, and must be supplied by the portable genius. Of course, the problem of the decision maker is to select the $d(n)$ that will yield the highest expected utility from the process. Therefore we have the recursive equation,

$$\begin{aligned} v_1(n) &= \text{Max}_{d(n)} \{p(A \mid d(n))[(\text{expected utility} \mid A, \\ &\qquad d(n)) + v_2(n-1)] + [1 - p(A \mid d(n))]v_1(n-1)\} \\ &= \text{Max}_{d(n)} \{p(u \geqq d(n))\langle u \mid u \geqq d(n)\rangle \\ &\qquad + [1 - p(u \geqq d(n))]v_1(n-1)\}, \qquad n = 1, 2, 3, \cdots . \end{aligned} \tag{47}$$

The value $v_1(0)$ of being in state 1 with no opportunities remaining must be specified as part of the problem statement. However, once $v_1(0)$ is known, Equation 47 can be used to find $v_1(n)$ for $n = 1, 2, 3, \cdots$ and thereby we can avoid using the portable genius.

We shall now specialize our results to the particular density function for u shown in Figure 4. It is clear that the probability that u will exceed $d(n)$ is just the area under this density function between $d(n)$ and 1 or $1 - d(n)$. The expected value of u given that u is greater than or equal to $d(n)$ is just halfway between $d(n)$ and 1 or $\frac{1}{2}[1 + d(n)]$. Therefore Equation 47 becomes simply

$$\begin{aligned} v_1(n) &= \text{Max}_{d(n)}\{[1 - d(n)]\tfrac{1}{2}[1 + d(n)] + d(n)v_1(n-1)\} \\ &= \text{Max}_{0 \leqq d(n) \leqq 1}\{\tfrac{1}{2}[1 - (d(n))^2] + d(n)v_1(n-1)\}, \\ &\qquad n = 1, 2, \cdots . \end{aligned} \tag{48}$$

The problem is now to find the value of $d(n)$ between 0 and 1 that maximizes the quantity in braces when $v_1(n-1)$ is known. Once more differentiation is an efficient procedure. We differentiate the quantity in the braces with respect to $d(n)$ and set the result equal to zero,

$$\begin{aligned} -d(n) + v_1(n-1) &= 0 \\ d(n) &= v_1(n-1), \qquad n = 1, 2, \cdots . \end{aligned} \tag{49}$$

From the second derivative we confirm that this solution represents a maximum rather than a minimum. We have therefore found that we should set the criterion level when n opportunities remain equal to the value of not having taken action when $n - 1$ opportunities remain: the form of the policy is very simple.

We find how well we shall do under this policy by evaluating the quantity in braces in Equation 48 at the value of $d(n)$ given by Equation 49,

$$\begin{aligned} v_1(n) &= \{\tfrac{1}{2}[1 - (d(n))^2] + d(n)v_1(n-1)\}_{d(n)=v_1(n-1)} \\ &= \tfrac{1}{2}[1 - (v_1(n-1))^2] + (v_1(n-1))^2 \\ &= \tfrac{1}{2}[1 + (v_1(n-1))^2], \qquad n = 1, 2, 3, \cdots . \end{aligned} \tag{50}$$

The expected utility to be derived from not having acted in a process when n opportunities remain is equal to $\frac{1}{2}$ of 1 plus the square of the expected utility when $n - 1$ opportunities remain, but only, of course, if the optimum policy specified by Equation 49 is followed.

Two special cases are of interest. The first is where $v_1(0) = 1$. This means that the highest utility is assured even if one never takes advantage of any of

TABLE 3

Results for a Uniform Distribution of Opportunity Utility

Opportunities Remaining n	Decision Criterion $d(n) = v_1(n-1)$	Value $v_1(n) = 0.5 + 0.5(v_1(n-1))^2$
0	—	0
1	0	0.50000
2	0.50000	0.62500
3	0.62500	0.69531
4	0.69531	0.74173
5	0.74173	0.77508
6	0.77508	0.80037
7	0.80037	0.82030
8	0.82030	0.83644
9	0.83644	0.84982
10	0.84982	0.86110
11	0.86110	0.87074
12	0.87074	0.87910
13	0.87910	0.88641
14	0.88641	0.89286
15	0.89286	0.89860
16	0.89860	0.90374

the opportunities. In this case Equations 49 and 50 show that

$$v_1(n) = d_1(n) = 1, \qquad n = 0, 1, 2, \cdots . \tag{51}$$

Because $d_1(n) = 1$, no action will ever be taken at any opportunity. Because $v_1(n) = 1$, the decision maker is assured of achieving the highest possible utility by following this policy. It is reassuring that even in this very unusual case the formulation provides the appropriate result.

However, a more interesting case arises when $v_1(0) = 0$. This value for $v_1(0)$ means that the decision maker is assured of the lowest possible utility if he does not take advantage of any opportunity. The values of $d(n)$ and $v_1(n)$ generated for this case using Equations 49 and 50 appear in Table 3. As we would expect, the criterion level when only one opportunity remains is 0—we are forced to act. However, the criterion level increases gradually with n, showing that we are getting more and more selective as the number of opportunities available to us increases. Furthermore, the expected utility that would be gained by following the optimum policy increases with the number of opportunities n. The rate of increase is greater when n is small than it is when n is large. The quantities $d(n)$ and $v_1(n)$ approach 1 and each other asymptotically as n becomes larger and larger. Thus we can become arbitrarily choosy about accepting an opportunity and very optimistic therefore about how well we shall do in expected utility provided that a large enough number of opportunities remain.

Note that the policy is very easy to implement. All we need is a card showing the decision criterion $d(n)$ for each n. The first time we encounter an opportunity whose utility equals or exceeds the corresponding $d(n)$, we act. Thus implementation would be practical even for the truck driver buying gasoline.

A Non-Optimum Policy

To convince ourselves that we have found the optimum policy for this problem, let us consider another policy: Act the first time the utility of an opportunity is better than average unless only one opportunity remains in which case act regardless of the utility presented. This policy requires

$$d(1) = 0; \qquad d(n) = 0.5, \qquad n = 2, 3, 4, \cdots . \tag{52}$$

We observe that Equation 48 allows us to evaluate any policy, even non-optimal ones, if we use the actual values of $d(n)$ rather than the optimizing values of Equation 49. We therefore have

$$v_1(n) = \tfrac{1}{2}[1 - (d(n))^2] + d(n)v_1(n-1), \qquad n = 1, 2, 3, \cdots . \tag{53}$$

Using the policy of Equation 52 we find

$$\begin{aligned} v_1(1) &= 0.500 \\ v_1(n) &= \tfrac{1}{2}[1 - (0.5)^2] + 0.5\, v_1(n-1) \\ &= 0.375 + 0.5\, v_1(n-1), \qquad n = 2, 3, 4, \cdots . \end{aligned} \tag{54}$$

Table 4 shows the values of $v_1(n)$ computed from Equation 54. We must remember that $v_1(n)$ is now the value of being in state 1 with n opportunities remaining when we are using the policy of Equation 52. By comparing Tables 3 and 4 we observe that when $n = 1$ or 2 the expected utility from the process under both policies and in fact the policies themselves are identical. However,

TABLE 4
Evaluation of a Non-Optimum Policy

Opportunities Remaining n	Decision Criterion $d(n)$	Value $v_1(n) = 0.375 + 0.5v_1(n-1)$
0	—	0
1	0	0.50000
2	0.5	0.62500
3	0.5	0.68750
4	0.5	0.71875
5	0.5	0.73438
6	0.5	0.74219
7	0.5	0.74610
8	0.5	0.74805
9	0.5	0.74903
10	0.5	0.74952
11	0.5	0.74976
12	0.5	0.74988
13	0.5	0.74994
14	0.5	0.74997
15	0.5	0.74999
16	0.5	0.75000
		⋮
		0.75

when n exceeds 2, then the expected utilities for the optimum policy shown in Table 3 always exceed the expected utilities for the present policy shown in Table 4 for the same value of n. Moreover, the asymptotic expected utility when n is large under the present policy is 0.75 rather than 1. The present policy does not provide the appropriate balance of birds in the hand and birds in the bush. Note, for example, that when 16 opportunities remain the decision maker will obtain an increase of about 0.15 in expected utility by following the optimum policy rather than the policy of Equation 52.

Grafting Optimum Policies

Suppose that for some reason we are forced to use a non-optimum policy for some portion of our sequential decision process. How should we make decisions beyond this unfortunate region? Imagine that the non-optimum policy of Equation 52 must be followed only when 8 or fewer opportunities remain. What should we do when more than 8 opportunities remain? We already know the answer. Equation 49 states that regardless of where $v_1(n - 1)$ came from, the best thing to do is to make $d(n) = v_1(n - 1)$. Table 5 shows the result of this idea. The portion of the table for $n \leqq 8$ is the same as the corresponding part of Table 4 for the non-optimum policy. However, when $n = 9$ we know that we should make $d(9) = v_1(8)$, no matter how $v_1(8)$ was produced. Since we are now operating under the optimum policy, $v_1(9)$ is computed from $v_1(8)$ using Equation 50. Thus the rows in Table 5 corresponding to $n = 9$ through 16 are obtained by applying Equations 49 and 50 from the point where the non-optimum policy

TABLE 5

Results for the Case of an Optimum Policy Grafted onto a Non-Optimum Policy

Opportunities Remaining n	Decision Criterion $d(n)$	Value $v_1(n)$	
0	—	0	
1	0	0.50000	
2	0.5	0.62500	Non-Optimum
3	0.5	0.68750	
4	0.5	0.71875	$v_1(n) =$
5	0.5	0.73438	$0.375 + 0.5\ v_1(n - 1)$
6	0.5	0.74219	
7	0.5	0.74610	
8	0.5	0.74805	
9	0.74805	0.77979	
10	0.77979	0.80404	Optimum
11	0.80404	0.82324	
12	$d_1(n) =$ 0.82324	0.83886	$v_1(n) =$
13	$v_1(n - 1)$ 0.83886	0.85184	$0.5 + 0.5\ (v_1(n - 1))^2$
14	0.85184	0.86282	
15	0.86282	0.87223	
16	0.87223	0.88039	

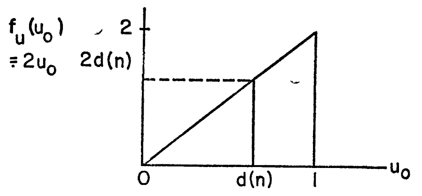


FIG. 5. A triangular density function for the utility presented at each opportunity

stopped. We observe that both $d(n)$ and $v_1(n)$ will once more approach 1 asymptotically when n is large.

However, it is important to note that the values for $d_1(n)$ and $v_1(n)$ in the rows from $n = 9$ through 16 in Table 5 are *not* the same as the corresponding policy and value numbers in Table 3 where the optimum policy could be followed throughout. The difference is not too large when $n = 16$, but it exists and is important. We could easily compute that using the values for $d_1(n)$, $n = 9, 10, \cdots, 16$ given by Table 3 would lead to lower values of $v_1(n)$ than do the values of $d_1(n)$, $n = 9, 10, \cdots, 16$ indicated in Table 5. It is just not true that if we are forced to use a non-optimum policy for the final stages of the decision process, then we should use for decisions beyond that point the same decisions we would have made if we had been able to follow an optimum policy throughout. However, the decision rule for the optimum policy does apply in this event; it serves well in computing the appropriate decision criteria to use in the range for n where the policy is allowed to be anything we please. The nature of a dynamic programming solution is perhaps better evidenced by the question of grafting on an optimum policy than by any other single topic we have discussed.

A Triangular Utility Distribution

We might be interested in how much the nature of our results depend on the uniform distribution of utility of Figure 4. Let us therefore suppose that the utility distribution is the triangular distribution of Figure 5. The effect of this change is to make higher utilities more prevalent than before. We still assume that the utilities presented at successive opportunities are independently selected from this triangular distribution. The two quantities that we need to provide in Equation 47 are the probability that u will equal or exceed $d(n)$ and the expected value of u given that it does. The probability that u will equal or exceed $d(n)$ is the area under the triangular probability distribution between $d(n)$ and 1, an area readily found to be $1 - (d(n))^2$. The density function for u conditional on u equalling or exceeding $d(n)$ is then $2\,u_0/[1 - (d(n))^2]$ for $d(n) \leqq u_0 \leqq 1$. Consequently the expected value of u given that u equals or exceeds $d(n)$ is

$$(55)\quad \begin{aligned}\langle u \mid u \geqq d(n)\rangle &= \int_{d(n)}^{1} u_0\, 2u_0/(1 - (d(n))^2)\, du_0 \\ &= \tfrac{2}{3}[1 - (d(n))^3]/(1 - (d(n))^2).\end{aligned}$$

When we substitute these results into Equation 47 we obtain

$$(56)\quad \begin{aligned} v_1(n) &= \text{Max}_{d(n)}\{[1 - (d(n))^2]\tfrac{2}{3}[1 - d(n))^3]/[1 - (d(n))^2] \\ &\qquad + (d(n))^2 v_1(n-1)\} \\ &= \text{Max}_{0 \leqq d(n) \leqq 1}\{\tfrac{2}{3}[1 - (d(n))^3] + (d(n))^2 v_1(n-1)\}, \\ &\qquad\qquad n = 1, 2, 3, \cdots. \end{aligned}$$

We observe that the denominator of the conditional expectation is just the probability that u equals or exceeds n. Therefore the contribution of the first term within the braces is just the integral of $u_0 f_u(u_0)$ from $d(n)$ to 1, a partial expectation. We again find the maximizing value of $d(n)$ by differentiating the quantity in braces with respect to $d(n)$ and setting the result equal to zero,

$$(57)\quad \begin{aligned} -2(d(n))^2 + 2d(n)v_1(n-1) &= 0, \\ d(n) &= v_1(n-1). \end{aligned}$$

We have found once again that the decision criterion $d(n)$ when n opportunities remain should be the expected utility to be derived from the process when $n - 1$ opportunities remain. The policy rule is exactly the same as before.

However, the computation of the value function $v_1(n)$ produces a different result. When we substitute the optimizing value of $d(n)$ from Equation 57 into the recursive equation 56, we find

$$(58)\quad \begin{aligned} v_1(n) &= \{\tfrac{2}{3}[1 - (d(n))^3] + (d(n))^2 v_1(n-1)\}_{d(n)=v_1(n-1)} \\ &= \tfrac{2}{3}[1 - (v_1(n-1))^3] + (v_1(n-1))^3 \\ &= \tfrac{2}{3} + \tfrac{1}{3}(v_1(n-1))^3. \end{aligned}$$

Thus the procedure for constructing $v_1(n)$ from $v_1(n-1)$ is changed from the one prescribed by Equation 50.

Table 6 shows the decision criteria and expected utilities computed from Equations 57 and 58 for the case of the triangular utility distribution. By comparing these results with those of Table 3 we find that for a given number of opportunities remaining the triangular distribution of utilities requires a higher decision criterion and also predicts a higher expected utility from the process. This is not surprising because in the triangular distribution case, higher utilities are more prevalent and therefore the decision maker can be more selective with the expectation of obtaining a better result.

A General Utility Distribution

Finding the same rule for establishing decision criteria for both the uniform and triangular distributions of utility presented leads us to inquire if this rule might not apply to any arbitrary utility distribution. We know that for any utility distribution $f_u(\cdot)$, the expected contribution to utility from acting now is just the partial expectation of the utility between $d(n)$ and 1. Furthermore,

TABLE 6

Results for a Triangular Distribution of Opportunity Utility

Opportunities Remaining n	Decision Criterion $d(n)$	Value $v_1(n) = \frac{2}{3} + \frac{1}{3}(v_1(n-1))^3$
0	—	0
1	0	0.6667
2	0.6667	0.7654
3	0.7654	0.8161
4	0.8161	0.8478
5	0.8478	0.8698
6	0.8698	0.8860
7	0.8860	0.8985
8	0.8985	0.9084
9	0.9084	0.9165
10	0.9165	0.9233
11	0.9233	0.9290
12	0.9290	0.9339
13	0.9339	0.9382
14	0.9382	0.9419
15	0.9419	0.9452
16	0.9452	0.9481

we shall obtain $v_1(n-1)$ with the probability we do not act, a probability that is the area of the utility density function between 0 and $d(n)$. Therefore, for a general utility density function we can write Equation 47 as

$$(59) \qquad v_1(n) = \text{Max}_{d(n)}\left\{\int_{d(n)}^{1} u_0 f_u(u_0)\,du_0 + v_1(n-1)\int_0^{d(n)} f_u(u_0)\,du_0\right\}.$$

We now find the optimizing value of $d(n)$ by differentiating the quantity in braces with respect to $d(n)$ and setting the result equal to zero. To perform this differentiation we must recall the rules for differentiating with respect to a variable appearing in the limits of an integral. We obtain

$$(60) \qquad \partial\{\ \}/\partial\, d(n) = -d(n)f_u(d(n)) + v_1(n-1)f_u(d(n)) = 0.$$

Therefore our policy rule is

$$(61) \qquad d(n) = v_1(n-1),$$

provided only that the density function is not zero at the point $d(n)$. Of course, when the density function is zero, then there is no possibility of a utility arising in that region and no difficulty can result. Therefore we have shown that regardless of the probability density function for the utilities presented at each opportunity, the optimum rule for establishing decision criteria is to make each decision criterion equal to the expected utility to be obtained from the process when one fewer opportunity remains.

Of course, when we consider Equation 61 for a while we see that the result is just what we would expect. If n opportunities remain and we pass up our present opportunity, then what we can expect in the future is $v_1(n-1)$. We would be silly to accept any utility for the present opportunity that was lower than $v_1(n-1)$, and equally silly to refuse any that was higher. Therefore we should establish our decision criteria just as Equation 61 prescribes.

We obtain an equation for the value $v_1(n)$ for a general utility distribution by evaluating Equation 59 at the value of $d(n)$ indicated by Equation 61.

$$v_1(n) = \int_{v_1(n-1)}^{1} u_0 f_u(u_0)\, du_0 + v_1(n-1) \int_0^{v_1(n-1)} f_u(u_0)\, du_0 . \tag{62}$$

This equation will not be very easy to evaluate if $f_u(\cdot)$ assumes a complicated form. This is just one instance of a result we often find in dynamic programming: it may be far easier to specify what to do than it is to determine how profitable following the best policy will be. We can generally calculate optimum policies more readily than value functions.

If we substitute the value of the lower limit $v_1(n-1)$ for the quantity u_0 in the integrand of the first integral in Equation 62, we develop an interesting inequality,

$$\begin{aligned} v_1(n) &\geqq \int_{v_1(n-1)}^{1} v_1(n-1) f_u(u_0)\, du_0 + v_1(n-1) \int_0^{v_1(n-1)} f_u(u_0)\, du_0 , \\ v_1(n) &\geqq v_1(n-1) \left[\int_{v_1(n-1)}^{1} f_u(u_0)\, du_0 + \int_0^{v_1(n-1)} f_u(u_0)\, du_0 \right], \\ v_1(n) &\geqq v_1(n-1) \left[\int_0^1 f_u(u_0)\, du_0 \right], \\ v_1(n) &\geqq v_1(n-1). \end{aligned} \tag{63}$$

Thus the value of having n opportunities remaining must be monotonically non-decreasing in n. This is a property we have observed in our examples and one that we would expect to hold for even a general density function of utility.

Generalizations

The action timing problem we have analyzed can be generalized in some ways at little cost and in others only by incurring great difficulty. For example, suppose that the density function for utility presented was different for each opportunity. This would cause almost no increase in complexity over the problem we solved. The decision rule would still be given by Equation 61. All that would change is that the utility density function necessary in Equation 62 for computing the value function would have to have a subscript to indicate that it was the utility density function for the nth opportunity. Otherwise the analysis is unchanged. Since the assumption of the same utility density function for all opportunities is the criticism most easily leveled at the model we considered, we are fortunate that changing this requirement causes no difficulty whatsoever.

However, if we desired to generalize the problem by allowing the utilities of successive opportunities to be not only uncertain, but dependent on one another (as might be the case, for example, if the gasoline service stations were having a price war), then we have made the problem considerably more difficult. The state of the system would have to specify not only whether or not the action had been taken, but also the utilities observed at the opportunities not taken in the past. There would have to be a strong incentive for solving this model before the effort could be justified.

Finally suppose that the number of opportunities remaining is not known exactly but is itself uncertain. The result in this case is simplicity itself. We already know how to compute the value $v(n)$ of having n opportunities remaining. All we do is weight this function with respect to the probability distribution we assign for the number of opportunities remaining to obtain an expected profit $\bar{v}$. If the opportunity currently presented has a higher utility than $\bar{v}$ then we accept it; otherwise we reject it. If no further opportunities are presented, then of course we are finished. However, if another opportunity appears then we construct a new distribution for the number of opportunities remaining and repeat the process. This new distribution will usually be the original distribution renormalized over one fewer opportunity. Thus the problem of uncertainty in the number of opportunities remaining introduces no new difficulty.

To summarize, we can usually allow a dynamic programming problem to be very general in the dependence of its variables upon the stage of the process without incurring any serious computational difficulty. As soon as the variables of the process become dependent from one stage to the next or when the concept of stage itself becomes confused, then we should expect stormy weather.

The Dynamic Programming Formulation

We began by describing the general theory of dynamic programming in rough terms and then proceeded to a pair of examples that indicated the types of problem to which the theory is applied. Now we are ready to present the subject of dynamic programming in more formal terms to establish the generality of the concept.

Suppose that we wish to describe a system by M state variables s_i. The composite state of the system at any time can then be represented by an M-component state vector $\mathbf{s}$ with components s_i,

$$\mathbf{s} = \{s_1, s_2, \cdots, s_M\}. \tag{64}$$

Depending on the model we construct, the components of the state variable could represent pressure or temperature, voltage or energy, investments or accounts receivable.

We can visualize every state vector $\mathbf{s}$ as a point in the M-dimensional state space $\mathcal{S}$ shown in Figure 6. The representation is unique: every state vector corresponds to a point in the state space and every point in the state space corresponds to a state vector. If the values of the state variables change, then the position of the corresponding state vector $\mathbf{s}$ in the state space changes.

Suppose now that we have available a box of "transformations". Each transformation when applied to a state vector will change it into another state vector

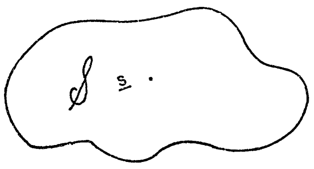


FIG. 6. The state space

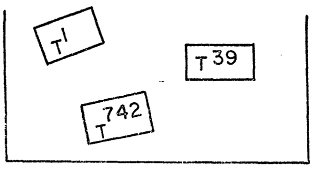


FIG. 7. The box of transformations

in the state space. A transformation in a physical problem might be an application of heat or the firing of a rocket. In a business problem it might represent a change in investment portfolio or the purchase of stock for inventory. Every transformation in the box is labeled with a number k for identification; thus T^k is the k^{th} transformation. If T^k is applied to the system when it is described by the state vector $\mathbf{s}$, the new state vector afterwards will be $T^k(\mathbf{s})$, another point in the state space $\mathcal{S}$. Figure 7 shows the box of transformations and a few transformations with their labels. The number of transformations in the box may be finite or countably infinite.

Suppose now that we choose n transformations from the box of transformations. We can choose the same transformation again and again as we like—there is no need that the transformations be different. Then we apply the n transformations in some order to the system. The transformations and their order could look like

$$T^{23}, T^{59}, T^{12}, T^{59}, T^{231}, \cdots, T^{77}. \tag{65}$$

We shall call the first transformation applied to the system T_1, the second T_2, etc. We must remember to distinguish between the identification number of a transformation and the number that indicates the order in which it is applied.

We shall specify that the system is originally described by the state vector $\mathbf{s}$, and denote its state vector after the first transformation T_1 is applied by $\mathbf{s}_1$, after the second transformation T_2 by $\mathbf{s}_2$, etc. The sequence of state vectors for the system therefore satisfies the successive equations,

$$\begin{aligned} \mathbf{s}_1 &= T_1(\mathbf{s}) \\ \mathbf{s}_2 &= T_2(\mathbf{s}_1) \\ \mathbf{s}_3 &= T_3(\mathbf{s}_2) \\ &\vdots \\ \mathbf{s}_n &= T_n(\mathbf{s}_{n-1}). \end{aligned} \tag{66}$$

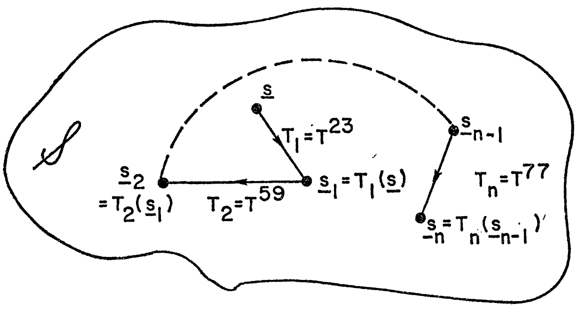


FIG. 8. Trajectory in state space

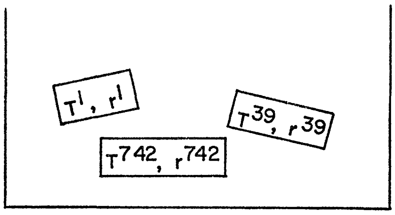


FIG. 9. The box of transformations with associated rewards

This sequence of state vectors establishes a trajectory in the state space, a trajectory shown in Figure 8. The particular trajectory that will be produced depends on the initial state vector **s**, the n transformations selected from the box, and the order in which they are applied.

To make trajectories interesting to us we assume that associated with the kth transformation in the box there is a reward function $r^k(\mathbf{s})$ that specifies the reward (money, fuel, power, etc.) that will be gained by applying the kth transformation to the system when it is described by the state vector **s**. Rewards may, of course, by either positive or negative. Each trajectory of the system will therefore have an associated set of rewards. For the particular trajectory we illustrated in Figure 8 the sequence of rewards is

$$(67) \qquad r^{23}(\mathbf{s}), \quad r^{59}(\mathbf{s}_1), \quad r^{12}(\mathbf{s}_2), \quad \cdots, \quad r^{77}(\mathbf{s}_n).$$

We denote this sequence by

$$(68) \qquad r_1(\mathbf{s}), \quad r_2(\mathbf{s}_1), \quad r_3(\mathbf{s}_2), \quad \cdots, \quad r_n(\mathbf{s}_{n-1})$$

in accordance with our notation for the sequence of transformations.

Figure 9 shows the box of transformations, each with an associated reward function. Figure 10 shows the system trajectory and the rewards generated by the application of each transformation. We now add one last feature to the model. We associate with each state vector **s** in the state space a terminal reward function $r_0(\mathbf{s})$ that specifies the additional reward to be gained if the system has state vector **s** after all transformations have been applied. We require such a function because in many problems no reward is paid while transformations are

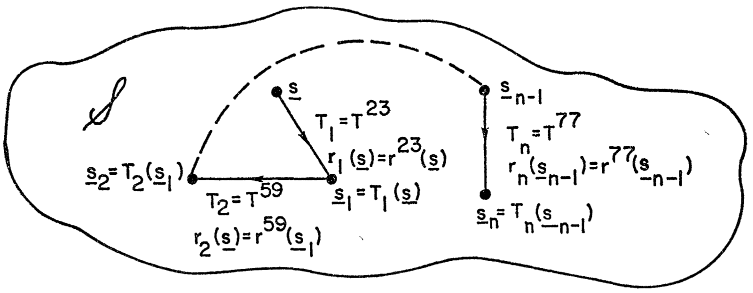

$\mathcal{S}$
$\underline{s}$
$T_1 = T^{23}$
$r_1(\underline{s}) = r^{23}(\underline{s})$
$\underline{s}_1 = T_1(\underline{s})$
$\underline{s}_2 = T_2(\underline{s}_1)$
$T_2 = T^{59}$
$r_2(\underline{s}) = r^{59}(\underline{s}_1)$
$\underline{s}_{n-1}$
$T_n = T^{77}$
$r_n(\underline{s}_{n-1}) = r^{77}(\underline{s}_{n-1})$
$\underline{s}_n = T_n(\underline{s}_{n-1})$

FIG. 10. The system trajectory with rewards

applied but only after they cease. We call these problems terminal value problems; they are good approximations to a broad class of control problems including the guidance of lunar probes. A more mundane need for terminal rewards is to include the effect of scrap value in machinery replacement problems.

Let us use P to indicate the policy represented by a particular set of transformations and a particular order of applying them. The total reward $r(\mathbf{s} \mid n, P)$ from being allowed to choose n transformations depends on the policy P and the starting state vector $\mathbf{s}$. Thus for a particular policy,

$$(69) \qquad r(\mathbf{s} \mid n, P) = r_1(\mathbf{s}) + r_2(\mathbf{s}_1) + \cdots + r_n(\mathbf{s}_{n-1}) + r_0(\mathbf{s}_n).$$

We can now imagine a game where the starting state vector $\mathbf{s}$ and the number of transformations allowed n are specified and we have to decide what transformations to use in what order so as to maximize the total reward from the trajectory that results. That is, we must find the best policy.

As we have said before, some of us will be better at this game than others and will be able to achieve higher total rewards. However, the maximum total reward that anyone can achieve is a function only of the starting state vector $\mathbf{s}$ and the number of transformations allowed n—these are the ultimate constraints on performance. Our portable genius knows how to achieve the maximum; we define the value function $v(\mathbf{s} \mid n)$ as the total reward he will obtain, the best anyone can do. Formally,

$$(70) \qquad v(\mathbf{s} \mid n) = \text{Max}_P\, r(\mathbf{s} \mid n, P)$$

where the maximization is carried out over all possible policies that can be constructed with n transformations.

Now we begin the solution for the best policy. If the system has state vector $\mathbf{s}$, we have n transformations to use and we use the kth transformation now, we shall obtain the reward $r^k(\mathbf{s})$ and produce a new state vector $T^k(\mathbf{s})$ for the system. Our problem is to find what it is worth to have the system described by a state vector $T^k(\mathbf{s})$ when we have $n - 1$ transformations left to use. We consult the portable genius; the answer is $v(T^k(\mathbf{s}) \mid n - 1)$. When we add this future reward to the present reward $r^k(\mathbf{s})$ we have obtained the total future reward to be expected from applying transformation k now. We examine all transformations k in this way to see which produces the highest total future re-

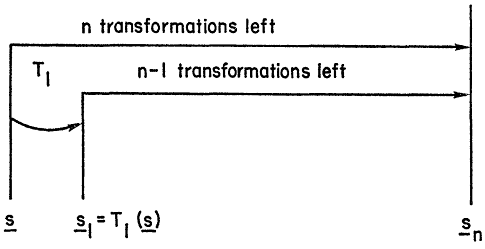


FIG. 11. The sequencing relationship

ward—the one that does is selected as the first transformation we apply. These results are summarized in the recursive equation,

$$v(\mathrm{s} \mid n) = \operatorname{Max}_k\{r^k(\mathrm{s}) + v(T^k(\mathrm{s}) \mid n-1)\}, \qquad n = 1, 2, 3, \cdots$$
$$T_1 = T^k; \qquad r_1(\mathrm{s}) = r^k(\mathrm{s}). \tag{71}$$

Figure 11 indicates the sequential nature of the result. If the value function is available when $n - 1$ transformations remain, we can find it when n remain. Of course, we must realize that this evaluation must be performed for every possible state vector in the state space. This computation can be prohibitively difficult if the number of possible state vectors is very large. However, at least theoretically Equation 71 shows us how to solve the problem if only someone will specify the function $v(\mathrm{s} \mid 0)$. Yet this is just the reward to be gained by having the system described by the state vector s when no transformations remain; therefore it is equal to the terminal reward,

$$v(\mathrm{s} \mid 0) = r_0(\mathrm{s}). \tag{72}$$

Thus we have been able to eliminate the necessity for the portable genius. The sequence of functions $v(\mathrm{s} \mid n)$ and the optimum policy are computable directly from the specifications of the problem.

This completes our formalization of the solution principle of dynamic programming. The principle is called the "principle of optimality" and results in recursive equations with the form of Equation 71. We see in the general formulation that the value function must always be unique, while the policy function need not be.

There is one last point we should mention on the general formalism. In some situations the results of applying transformations to the system may not be deterministic. Rather the new state vector, the reward generated, or both may have to be described by random variables. However, we can define an expected total future reward $\bar{v}(\mathrm{s} \mid n)$ by analogy with Equation 70 and then realize that since the expected value of a sum is the sum of the expected values, we can write a recursive relation for $\bar{v}(\mathrm{s} \mid n)$ directly from Equation 71,

$$\bar{v}(\mathrm{s} \mid n) = \operatorname{Max}_k\{\bar{r}^k(\mathrm{s}) + \bar{v}(T^k(\mathrm{s} \mid n-1))\}, \qquad n = 1, 2, 3, \cdots$$
$$T_1 = T^k; \qquad \bar{r}_1(\mathrm{s}) = \bar{r}^k(\mathrm{s}). \tag{73}$$

Here we have used $\bar{r}^k(\mathbf{s})$ to designate the expected reward received by applying transformation k to the system when it is described by state vector $\mathbf{s}$. To allow for the possibility that the terminal reward $r_0(\mathbf{s})$ may be a random variable, we replace Equation 72 by

$$\bar{v}(\mathbf{s} \mid 0) = \bar{r}_0(\mathbf{s}). \tag{74}$$

Equations 73 and 74 now serve to find the policy that maximizes the expected value of total future reward. However, since the solution rests on the linearity of the expectation operator, we should not find it surprising that using other criteria (as in finding the policy with the minimum reward variance) may be very difficult.

Conclusion

We have now seen both the general structure of dynamic programming and its application to two fairly representative examples. From our discussion it is clear that applying dynamic programming requires considerable insight on the part of the analyst if he is to avoid creating a computationally infeasible problem. Therefore, dynamic programming is an approach most likely to be used by the professional analyst rather than by a manager directly. This situation stands in contrast to the widespread usage of linear programming.

As we can see from our examples, the number of points that must be evaluated in the recursive procedure increases rapidly with the number of state variables and the number of levels each state variable can assume. If there are a state variables, each assuming b levels, then we must consider b^a points in the state space. This number clearly becomes prohibitive very quickly for even the largest computers. By being clever in the evaluation process we can often reduce the amount of computation below the above requirements; however, any major reduction requires further assumptions about the structure of the problem.

One assumption on problem structure that produces computational simplifications is that the optimum trajectory is a continuous or nearly continuous function in the state space. This means that regardless of which transformation is applied to the system when it has a certain state vector, the state vector resulting from the transformation will be close in some sense to the original state vector. The assumption is most often justified in physical as opposed to business control systems. For example, an airplane subject to drag and propulsive forces will change its state arbitrarily little if the time for the change is small enough. However, an inventory system subject to large batch orders can, at least on paper, experience a sudden drastic change of state. Yet even in systems whose trajectories do contain discontinuities we can often identify regions of continuous operation. The continuity assumption simplifies computation because it limits the number of possible state vectors that must be evaluated.

Several attempts have been made to construct general dynamic programming formulations that were suitable as models for broad classes of even discontinuous problems and yet retained computational feasibility. The most important formulation of this type is the decision model based on the Markov process. In this formulation we model the problem as a finite-state Markov process that earns rewards as it makes transitions from state to state. The reward earned can be particular to the transition made. If the process is allowed to make a very large number of transitions, then it will earn some average amount of reward per transition, a number that we call the gain of the process.

The opportunity for making decisions arises whenever the process enters a state. We can select from a number of alternatives the alternative we want to govern the system's behavior until it leaves that state. Each alternative specifies both the probabilistic structure and the reward structure that the system will be subject to until it enters another state. The most usual decision problem in such a process is to determine which alternatives should be used in each state so as to maximize the gain of the process. Special algorithms exist for solving this problem; moreover, some linear programming techniques are suitable for performing the computation.

The dynamic programming formalism based on the Markov process has found application in a wide variety of practical situations. These include maintenance and repair, financial portfolio balancing, inventory and production control, equipment replacement, and directed marketing. The list grows steadily as more and more enterprises see the advantage of a formal representation of their sequential decision problems. The Markovian formulation is a natural starting point because its computability is assured.

Thus dynamic programming is a concept that gains practical implications as its domain of application is narrowed. It shares this property with all other mathematical concepts that have ever been proposed. The challenge of dynamic programming is thus a challenge to our ability to apply a very interesting concept to the solution of practical problems.

References

1. Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.
2. —— and Dreyfus, S., *Applied Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1962.
3. Howard, R., *Dynamic Programming and Markov Processes*, M. I. T. Press, Cambridge, Massachusetts, 1960.