Introduction To Dynamic Programming

George L. Nemhauser
Georgia Institute of Technology
313 PUBLICATIONS **22,656** CITATIONS

JSS 16 (4) (1961) 261-274

AN INTRODUCTION TO DYNAMIC PROGRAMMING

by

BRIAN GLUSS

(*Armour Research Foundation of Illinois Institute of Technology*)

DYNAMIC programming, a mathematical field that has grown up in the past few years, is recognized in the U.S.A. as an important new research tool. However, in other countries, little interest has as yet been taken in the subject, nor has much research been performed. The objective of this paper is to give an expository introduction to the field, and give an indication of the variety of actual and possible areas of application, including actuarial theory.

In the last decade a large amount of research has been performed by a small body of mathematicians, most of them members of the staff of the RAND Corporation, in the field of multi-stage decision processes, and during this time the theory and practice of the art have experienced great advances. The leading force in these advances has been Richard Bellman, whose contributions to the subject, which he has entitled *Dynamic Programming*[1], have had effects not only in immediate fields of application but also in general mathematical theory; for example, the calculus of variations (see chapter IX of [1]), and linear programming (chapter VI). The work done so far has paved the way for innumerable further research and applications, and hence the field, besides being a very stimulating one, is one which new blood will find potentially highly fruitful.

As will be seen when the concept of dynamic programming is defined and illustrated more fully below, a further development in the last decade that has made research in the subject more feasible and useful has been the evolution of the digital computer, which has made it possible to simulate complicated processes and hence deduce their properties by Monte Carlo methods; sometimes, also, simulation has provided hints as to the solution of thorny mathematical problems that have arisen out of the study of the processes.

THE TERM 'DYNAMIC PROGRAMMING'

Decision processes exist in which it is required to make just one decision which will completely determine the outcome of the process. For example, we may bet on the throw of a die and, of course, if we have observed previous tosses from which we have deduced the die is loaded, our decision may not necessarily be arbitrary. However, a more interesting type of decision process is a multi-stage one; that is, one in which a sequence of decisions must be made, each of which will affect the final outcome of the process. Also, in such a process the order of the decisions may be crucial. An example, of a stochastic nature, is an allocation process in which funds on hand at one point in time must be distributed among various investments, and then the returns re-distributed if necessary at each point of time so as to maximize the expected final return; information, such as seasonality and trends, will naturally affect our decisions. Another example, of a deterministic nature, is the writing of a computer program, in which the order of instructions—or decisions—is of course of paramount importance.

The fact that in most of these multi-stage decision processes time plays a significant role accounts for the term 'dynamic'. Of course, this does not preclude the possibility of embedding other processes in a dynamic programming formulation, and in fact many procedures evolved in the subject provide new approaches to processes in which time plays no part.

The word 'programming' is used not in the restricted computer sense but in that now encompassed by the phrase mathematical programming. That is, we wish to describe, in formal mathematical terms, a system or process and then consider the mathematical model thus obtained. The word programming implies not only the translation of the system to a mathematical model but also the kind of model, the kind of mathematical functions we choose to manipulate, and how we manipulate them.

BASIC PROPERTIES

The basic characteristic of problems in dynamic programming is that they are formulated as models involving N-stage decision processes, where the desired solution is the determination of 'policies' or 'strategies' which satisfy criteria, which, for example, could be minimization of cost, or, in stochastic models, of the statistical expectation of the cost or maximization of profit. To avoid any confusion with insurance terminology, the term 'strategy' will be used throughout this paper. One of the basic techniques used is to reduce the N-dimensional optimization problem to N sequential one-dimensional problems using Bellman's so-called 'Principal of Optimality':

'*An optimal strategy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal strategy with regard to the state resulting from the first decision.*'

These one-dimensional problems lead to sequential equations which are often amenable to mathematical iterative solution, and even when this is not so, the iterative equations often prove much simpler to handle on a digital computer for specific solutions than the original N-dimensional optimization problem, reducing considerably the amount of storage required.

An example of this type of reduction—applied to a problem independent of time—is as follows

Suppose it is desired to find the maximum of

$$F(x_1, \ldots, x_N) = \sum_1^N g_i(x_i) \tag{1}$$

over the segment of the plane

$$x_1 + \ldots + x_N = c; \quad x_i \geqslant 0, \tag{2}$$

where the g_i are continuous. Since the maximum depends only upon c and N, define it to be $f_N(c)$.

Then, using the principle of optimality,

$$f_N(c) = \max_{0 \leqslant x_N \leqslant c} [g_N(x_N) + f_{N-1}(c - x_N)]. \tag{3}$$

That is, the maximum equals the maximum of the sum of (i) the last term of equation (1) and (ii) the maximum of the first $N-1$ terms of F subject to the constraint $x_1+\dots x_{N-1} = c-x_N$; $x_i \geqslant 0$ in place of equation (2).

Also, $f_1(c) = g_1(c)$, and hence one may solve successively for $f_2, f_3, \dots, f_N$. For example,

$$f_2(c) = \max_{0 \leqslant x \leqslant c} [g_2(x)+g_1(c-x)],$$

where, since g_1 and g_2 are mathematically defined, the value of x that maximizes the function in parentheses is easily determined by calculus. Whether this value of x is within the interval $(0, c)$ or at one of its boundaries $0, c$, this value x'—which is a function of c alone—is inserted in the function in parentheses, and $f_2(c)$ is given by

$$f_2(c) = g_2(x')+g_1(c-x')$$

$f_3(c)$ is then obtained in exactly the same way, using equation (3) with $N=3$; and so on.

Note that this kind of technique of maximization eliminates the trouble that plagues such well-known techniques as partial differentiation and Lagrangian multipliers; that is, that if the solution happens to be on a boundary, as so often happens in maximization problems (cf. for example, linear programming), then the equations break down. This trouble is never encountered in Bellman's maximization procedures.

The crux of dynamic programming formulation, then, is to define functions $f_N(x)$—or in multi-dimensional cases $f_N(x_1, \dots, x_r)$—so that they may be expressed in terms of $f_i(x)$, $i < N$. The most important part of the procedure is to choose the 'right' function so that such equations may be developed and solved, and this to a great extent is a matter of experience and intuition, which may be partially acquired by delving through the literature. Reference [1] is an obvious starting point towards this end, and the bibliography in it gives ample further references.

The following example comes from reference [1], (example 45, chapter I) and is chosen because of its similarity to valuation problems. The notation has been slightly amended.

Example (i)

Suppose that we have a machine whose output in time periods 1, 2, 3, ... is $r_1, r_2, r_3, \ldots$, and its upkeep cost $u_1, u_2, u_3, \ldots$. The purchase price of a new machine is p, and its trade-in value at the end of the tth period is s_t. $p > s_0$, where s_0 is defined as the trade-in value at the beginning of the first period. The discounting factor is v.

Let the present value of future returns using an optimal strategy from the end of the tth period $= f_t$, where f_0 is defined as at the beginning of the first period but immediately after purchase.

Assume r_t and u_t are to be discounted from the end of the period.

The recurrence relationships are

$$f_0 = \max \begin{bmatrix} v(r_1 - u_1 + f_1) \\ s_0 - p + f_0 \end{bmatrix}$$

$$= v(r_1 - u_1 + f_1) \quad \text{for practical purposes}$$

and $$f_t = \max \begin{bmatrix} v(r_{t+1} - u_{t+1} + f_{t+1}) \\ s_t - p + f_0 \end{bmatrix}.$$

The recurrence relationship gives

$$f_0 = v(r_1 - u_1 + f_1),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$

$$f_{n-1} = v(r_n - u_n + f_n),$$

$$f_n = s_n - p + f_0$$

if n is the value of t for which

$$s_t - p + f_0 > v(r_{t+1} - u_{t+1} + f_{t+1}).$$

These equations give

$$f_n = \frac{f_0}{v^n} + \frac{(u_1 - r_1)}{v^{n-1}} + \ldots + (u_n - r_n)$$

and $$f_n = s_n - p + f_0.$$

Eliminating f_n,

$$v^n(s_n - p + f_0) = f_0 + v(u_1 - r_1) + \ldots + v^n(u_n - r_n),$$

i.e. $$f_0 = \frac{v^n s_n - p - [v(u_1 - r_1) + \ldots + v^n(u_n - r_n)]}{1 - v^n}.$$

Hence to maximize the present value of future returns, n must be chosen to maximize this expression.

Example (ii)

Reference [1] chapter I, example 47, asks further if it is uniformly true that the optimal policy, if given an over-age machine, is to turn it in immediately for a new one.

Consider the value of the process again:

If we are given a machine aged T and sell it immediately, then the value of the process is

$$s_T - p - f_0.$$

If we save it one time-period the value is

$$v(s_{T+1} - p + f_0 + r_{T+1} - u_{T+1}).$$

Hence, if

$$v(r_{T+1} - u_{T+1}) > s_T - v s_{T+1} + (f_0 - p)(1 - v),$$

then we should not trade it in immediately. This of course will only be the case when the return per time-period on the total profit $f_0 - p$ at the rate of interest inherent in v is smaller than the difference between the return r_{T+1} and the upkeep u_{T+1}. This is seen more easily if we put $v = 1/(1+i)$ and write the inequality as

$$(r_{T+1} - u_{T+1}) > (1+i)\, s_T - s_{T+1} + (f_0 - p)\, i.$$

Example (iii)

Reference [1], chapter I, example 48, asks how one would formulate the problem to take into account technological improvement in machines and operating procedure.

Here the inclusion in the recurrence relationship of a penalty for outdating would be reasonable, as a loss is incurred due to the inefficiency of the old machine relative to a new type. Hence the equation would read something like

$$f_t = \max \begin{bmatrix} v(r_{t+1} - u_{t+1} - g_{t+1} + f_{t+1}) \\ s_t - p + f_0 \end{bmatrix}.$$

Outdating would also affect s_t but it is assumed this is incorporated in s_t itself.

The expression arrived at in example (i) could be obtained directly by standard actuarial techniques:

If sold after the nth period every time the value

$$=\{-p+v(r_1-u_1)+\dots+v^n(r_n-u_n)+v^n s_n\}\,\{1+v^n+v^{2n}+\dots\}.$$

Define commutation functions

$$C_t=v^t(r_t-u_t)\quad(t\neq 0),$$
$$C_0=-p,$$
$$D_n=v^n s_n$$

and

$$M_n=C_0+C_1+\dots+C_n,$$

all of which can be tabulated.

Then value

$$=(M_n+D_n)\,(1+v^n+v^{2n}+v^{3n}+\dots)$$
$$=\frac{M_n+D_n}{1-v^n},$$

which can be tabulated to obtain the maximum value by inspection.

The solution to example (ii) might also suggest to the actuarial student that he would more naturally pose the original problem as one of maximizing the inherent rate of interest i.

Thus for an initial outlay of p the future returns are

$$r_1-u_1,$$
$$\dots\dots\dots\dots,$$
$$r_n-u_n+s_n-p,$$
$$r_1-u_1,$$
$$\dots\dots\dots\dots,$$

etc. in successive periods.

Hence the inherent rate of interest i is given by

$$p=v(r_1-u_1)+\dots+v^n(r_n-u_n-s_n-p)+v^{n+1}(r_1-u_1)+\dots,$$

where

$$v=1/(1+i).$$

It would be required to find the value of n which maximized i.

SUCCESSIVE APPROXIMATION IN STRATEGY SPACE

Another important concept in dymanic programming involves the use of successive approximation in solving complicated functional equations of the type

$$f_N(x)=\max_{0\leqslant x_{N-1}\leqslant x}[T(f_{N-1},x_{N-1})].\tag{4}$$

An equation of this kind is equation (3) when $g_N(x) = g(x)$, all N. Now suppose that the process is unbounded in time. We then wish to solve the asymptotic equation

$$f(x) = \max_{0 \leqslant X \leqslant x} [T(f, X)]. \tag{5}$$

Solution of this equation comprises finding the values of X, i.e. the strategies, and of $f(x)$, the outcomes of the process, for all x. Sometimes exact solutions are easily obtainable; however, often they are not. In such cases, a powerful procedure for obtaining successive approximations to the solution is to make successive approximations in *strategy space* rather than function space; in other words, to make successive approximations $X^{(1)}, X^{(2)}, \ldots$ to X, where these $X^{(i)}$ are of course all functions of x. First, we assume a solution $X^{(1)}$, and then solve the equation

$$f(x) = T(f, X^{(1)}) \tag{6}$$

to obtain a first approximation $f^{(1)}(x)$ to $f(x)$. Then $X^{(2)}$ is obtained by solving

$$f^{(1)}(x) = \max_{0 \leqslant X^{(2)} \leqslant x} [T(f^{(1)}, X^{(2)})], \tag{7}$$

and so on, finding alternately successive values $f^{(2)}, X^{(3)}, f^{(3)}, X^{(4)}, \ldots$ of the function and the strategy.

The powerful nature of this concept is obvious, in that it may not only be used as a purely mathematical technique but also as one partially using computer simulation of the process. This will be discussed further after consideration of an example of the application of dynamic programming and successive approximation in strategy space to an actuarial problem, which might clarify concepts slightly, even though the model might be considered somewhat oversimplified.

Example (iv)

Suppose an insurance company has a certain number of life insurance policies and must decide how large its liquid reserves should be in order to handle all claims. If the reserves are too large, the company is losing money by failing to invest the surfeit; if they are too small, additional costs are incurred in meeting the excess claims, for example, by being forced to sell securities under unfavourable circumstances. What, then, is the optimal reserve level to adopt?

Let it be assumed that if there are reserves x on hand at time t, we wish to determine what additional reserves $y-x$ should be put aside in the next time-period $(t, t+1)$ for payment of claims, given that the probability distribution for the amounts of claims for the time-period is $\phi(s)\,ds$, that it costs $k(y-x)$ to make these additional reserves available (for example, in administrative costs and lost investment dividends), that a cost $p(z)$ is incurred if there is an excess z of claims over reserves, and that there is a discount ratio v, $0 < v < 1$, per unit time.

Then, if $f(x)=$ expected total discounted cost of the process starting with initial reserves x and using an optimal policy,

$$f(x) = \max_{y \geqslant x}\left[k(y-x)+v\int_y^\infty p(s-y)\,\phi(s)\,ds+vf(0)\int_y^\infty \phi(s)ds\right.$$
$$\left.+v\int_0^y f(y-s)\phi(s)\,ds\right]. \tag{8}$$

The first term is the cost of adding reserves $y-x$; the second is the integral of the probability that $s>y$ and a cost $p(s-y)$ is incurred; the third term is the cost of the process from the next time-period given that reserves have been completely exhausted (i.e. $s \geqslant y$); and the fourth term is the integral of the probability that $s<y$ and we are left at time $t+1$ with reserves $y-s$, from when the cost of the process is $f(y-s)$.

This is precisely equation (7), p. 156, of [1], and the reader is referred to it for a discussion of the solution. An exact solution is given for $k(z)=kz$ and $p(z)=pz$; further cases are considered there and also in references [2] and [3]. The optimization criterion used was that of minimizing $f(x)$. It turns out, incidentally, that the exact solutions in some of the cases just referred to are constant reserve level solutions. That is, if at time t our reserves are x, we order reserves $y-x$, where y is independent of x and depends only upon the parameters of the system. This would appear to be actuarially convenient.

For example, if $\phi(s)\,ds$, the distribution for the amount of claims,

$$= e^{-s}\,ds,$$

and the cost $p(z)$ of an excess z of claims over reserves

$$= pz$$

and k is also taken as a constant, then

$$f(x) = \min_{y \geqslant x} \left[k(y-x) + v\int_y^\infty p(s-y)\, e^{-s}\,ds + vf(0)\int_y^\infty e^{-s}\,ds + v\int_0^y f(y-s)\,e^{-s}\,ds \right]$$

$$= \min_{y \geqslant x} \left[k(y-x) + vp\, e^{-y} + vf(0)\, e^{-y} + v\int_0^y f(y-s)\, e^{-s}\,ds \right].$$

On p. 160 it is given that if $vp > k$, and some other simple practical conditions are also satisfied, then the solution is as follows:

Let $\bar{x}$ be the unique root of

$$k = vp\int_y^\infty e^{-s}\,ds + vk\int_0^y e^{-s}\,ds,$$

i.e. of $$k = vp\, e^{-y} + vk(1-e^{-y}),$$

i.e. $$e^y = \frac{v(p-k)}{k(1-v)},$$

i.e. $$\bar{x} = \log\frac{v(p-k)}{k(1-v)}.$$

Then the optimal policy has the form

(i) for $0 \leqslant x \leqslant \bar{x}$ $(y = \bar{x})$,

(ii) for $x \geqslant \bar{x}$ $(y = x)$.

In other words, the optimal reserve level is $\bar{x}$.

If $vp \leqslant k$, the solution is given by $y = x$ for $x \geqslant 0$, i.e. never order.

This is intuitively obvious since $vp < k$ implies that the cost of a penalty is less than the cost of ordering its equivalent in advance.

Of course, we could alternatively consider 'suboptimal' strategies. For example, we could confine ourselves to constant reserve level solution and deduce from equation (8) what the constant level should be. That is, solutions would be confined to a subset of possible solutions, and the optimal solution of this subset determined.

Also, for functions $k(z)$ and $p(z)$ that make equation (8) difficult to solve exactly, successive approximations in strategy space may be found in the manner described in the previous section.

SIMULATION OF MULTI-STAGE DECISION PROCESSES

A powerful new tool for investigating complicated multi-stage decision processes is the digital computer: successive approximations to optimal strategies may be derived by means of simulation of the processes on the computer. Using a specific strategy, and random numbers if the process is stochastic, the process may be performed again and again, after feeding the parameters and constraints of the system into the computer, and the outcomes observed. When this has been done a sufficiently large number of times, a clear picture of the results obtained from using the strategy will be evident: we will have a close approximation to the probability distribution of the outcomes. This method is generally referred to as a Monte Carlo method.

Once the outcomes have been observed, the strategy is changed slightly and a new set of outcomes is observed. If these are better with respect to the optimization criteria—for example, the average cost of a search process may have been reduced—then we now have a closer approximation to the optimal strategy. Hence a systematic technique for successive approximation in strategy space may be defined for any multi-stage decisions process.

As an example, suppose that it is required to find an object which is in one of N cells with probabilities $p_1, \ldots, p_N$, where $\sum_1^N p_i = 1$. Also, let there be corresponding cost parameters $t_1, \ldots, t_N$. Let it be assumed further that if a cell is searched and the object is in a neighbouring cell it will move one cell further away, unless it is in either of the cells 1, N, when it will remain where it is. Now, in the process in which no movement is permitted, it has been determined that the strategy that minimizes the statistical expectation of the total cost of the search is to examine in order of increasing values of t_r/p_r. This is an intuitively reasonable solution, since the procedure involves examining cells with low cost and high probability first.

Hence let us use this strategy as a first approximation to our more complicated process, and simulate the process a large number of times, noting what the costs of the search are. For successive approximations, consider the subset of possible strategies defined by minimizing $t_r/(p_r)^{k_r}$, where the k_r are initially set equal to unity, and then varied incrementally (and independently). For example, searching near the boundaries will have a different effect from searching elsewhere, and we may therefore decide to see what happens when we vary the k_r for r near 1 and N first. Hence the whole problem of determining the strategy in this subset may be (computer) programmed from beginning to end to simulate the process, change the k_r incrementally, observe when these changes produce positive and negative effects (i.e. decrease and increase the cost), and increment and decrement accordingly until the suboptimal strategy is determined. Another subset of strategies could comprise criteria of the form $k_r t_r/p_r$, for example. An interesting research problem for the reader, incidentally, would be to attempt to obtain the mathematically exact optimal strategy, which has not as yet been found. More sophisticated methods exist of applying the concept of successive approximation using Monte Carlo simulation on a computer; the example above and the strategy subsets considered were chosen to attempt to explain the concept as simply as possible.

FIELDS OF APPLICATION

The number of areas of application, actual and potential, of dynamic programming is extremely large, one might say virtually inexhaustible; and much research of considerable importance is being attacked successfully from this new angle. Probably one of the most important and promising of these areas is the field of learning processes [4], feed-back processes, and adaptive control processes [5], [6]. Much thought has been given in the last decade to the construction of machines that can 'learn'. The definition of this phrase as far as machines are concerned is to some extent a philosophical problem; simply, what we usually mean is to program the machine, or give it sufficient capacity, instructions and logic, to enable it to learn from its own experiences, and adapt its decisions, or strategies, according to the criteria for success that we feed into it. Hence, as it observes its successes and failures using certain strategies, it will modify them to increase the success rate, hoping eventually to achieve complete success. This is obviously possible with a computer of infinite storage space, because the totality of experiences may be stored to cite in future experiences; however, since the machine cannot have infinite storage space, other concepts, of a probabilistic nature, have to be developed.

Another important area is that concerning search processes, in which we may be considering a system in which we wish to find an object such as the cause of a breakdown in the system, for example [7], [8], or even find it and automatically repair it [9]. There naturally are also applications to war games in which it is desired to derive optimal strategies for detecting enemy objects, such as submarines. A further application is the retrieval of information from large libraries of data or documents. A simple example of an information retrieval problem is given on pp. 50–51 of [1].

Also, inventory control, and scheduling problems, as previously cited, have been considered with success from the dynamic programming viewpoint.

Dynamic programming has also been applied to such diverse fields as communication theory (see pp. 140–143 of [1], [10]), allocation processes (chapter I of [1]), and communication network theory [11], [12].

It should of course be realized that some of the fields and references mentioned are to a certain extent overlapping, depending upon one's definitions of these respective fields. However, the variety of applications should nevertheless be obvious, and it is to be hoped that this introduction to dynamic programming may attract further research workers into the field.

REFERENCES

[1] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press. (The reader should perhaps be cautioned that, as is to be expected for any important mathematical first edition, there are a certain number of slight errors.)

[2] Gluss, B. (1960). 'An optimal inventory solution for some specific demand distributions'. *Naval Research Logistics Quarterly*, 7, no. 1, 45–48.

[3] Levy, J. (1959). Further notes on the loss resulting from the use of incorrect data in computing an optimal policy. *Naval Research Logistics Quarterly*, **6**, no. 1, 25–31.

[4] Bellman, R. & Kalaba, R. (1958). On communication processes involving learning and random duration. *RAND Research Memorandum* P-1194.

[5] Bellman, R. (1961). *Adaptive Control; A Guided Tour*. Princeton University Press.

[6] Kalaba, R. (1959). Some aspects of adaptive control processes. *RAND Research Memorandum* P-1809.

[7] Johnson, S. M. (1956). Optimal sequential testing. *RAND Research Memorandum* RM-1652.

[8] Gluss, B. (1959). An optimum policy for detecting a fault in a complex system. *Operations Research*, 7, no. 4, 467–477.

[9] Firstman, S. I. & Gluss, B. (1960). Optimum search routines for automatic fault location. *Operations Research*, **8**, no. 4, 512–523.

[10] Bellman, R. & Kalaba, R. (1957). On the role of dynamic programming in statistical communication theory. *IRE Transactions of the Professional Group on Information Theory*, vol. IT-3, no. 3.

[11] Kalaba, R. & Juncosa, M. L. (1956). Optimal design and utilization of communication networks. *RAND Research Memorandum* P-782.

[12] Kalaba, R. (1959). On some communication network problems. *RAND Research Memorandum* P-1325.