An Introduction to Dynamic Programming

M. G. Simpson

Applied Statistics, Volume 10, Issue 1 (Mar., 1961), 32-38.

Your use of the JSTOR database indicates your acceptance of JSTOR's Terms and Conditions of Use. A copy of JSTOR's Terms and Conditions of Use is available at http://www.jstor.org/about/terms.html, by contacting JSTOR at jstor-info@umich.edu, or by calling JSTOR at (888)388-3574, (734)998-9101 or (FAX) (734)998-9113. No part of a JSTOR transmission may be copied, downloaded, stored, further transmitted, transferred, distributed, altered, or otherwise used, in any form or by any means, except: (1) one stored electronic and one paper copy of any article solely for your personal, non-commercial use, or (2) with prior written permission of JSTOR and the publisher of the article or other text.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

Applied Statistics is published by Royal Statistical Society. Please contact the publisher for further permissions regarding the use of this work. Publisher contact information may be obtained at http://www.jstor.org/journals/rss.html.

Applied Statistics
©1961 Royal Statistical Society

JSTOR and the JSTOR logo are trademarks of JSTOR, and are Registered in the U.S. Patent and Trademark Office. For more information on JSTOR contact jstor-info@umich.edu.

©2000 JSTOR

http://www.jstor.org/
Wed Mar 15 21:58:05 2000

AN INTRODUCTION TO DYNAMIC PROGRAMMING*

M. G. SIMPSON

In the course of their work statisticians are frequently called on to make the acquaintance of, and possibly to use, techniques from other fields, such as operational research. One such technique is Dynamic Programming; in this expository article Dr Simpson explains it in simple terms.

THIS paper is mainly intended to map out the general field of application of dynamic programming and to illustrate its basic concept, but the computation required is also indicated.

I must first make it clear that dynamic programming has little in common with the better-known technique of linear programming. It is true that there are many problems which may be solved using either technique, but in such cases it is nearly always preferable to use linear programming with its much simpler computation. Dynamic programming may, however, be applied to a wide range of problems to which linear programming is not applicable; other advantages of using the technique will be indicated later.

Multi-Stage Decision Processes

The technique was devised initially (Richard Bellman, *Dynamic Programming*, Oxford University Press, 1957) for the solution of a class of problems called multi-stage decision processes. By this we mean processes where decisions are taken sequentially and where they are not independent, in that decisions taken early on in the process affect decisions made subsequently. For example, if we consider a retailer who restocks with a certain commodity at, say, monthly intervals, then the amount which he orders in one month will certainly affect the amount he orders in the succeeding months. For if he orders too large an amount of stock he will probably require to order little if any the following month, whereas if he places too small an order initially he will run out of the commodity during the month and will have to order a substantial amount next time. In this we can see that the decisions made affect decisions to be made subsequently. This is the most straightforward class of multi-stage decision processes, in which the decisions are made sequentially in time. Although processes of this type arise sufficiently frequently for a technique for handling them to be worth while, it is also possible to tackle other problems in a similar manner. In certain problems, even if the time element is not explicit,

* Based on a paper read at a one-day conference on Dynamic Programming organised by the Birmingham Group of the Royal Statistical Society, May 1960.

it is sometimes advantageous to treat them as multi-stage processes and to use dynamic programming to solve them. An example which we shall consider in some detail is of this nature.

Characteristics of Multi-Stage Decision Processes

The types of processes which we can tackle using dynamic programming can be sub-divided in several ways.

For instance we can deal with problems which are either deterministic or stochastic. In deterministic processes the result of any decision is unique and is known beforehand. Stochastic processes are those in which the result of any decision is not known uniquely but can only be forecast in terms of some probability distribution—the retailer mentioned above is usually in this unfortunate stochastic state.

We can also consider problems in which the number of decisions available at each stage in the process is either finite or infinite. A 'finite' problem may have only two decisions possible at any stage, whereas an 'infinite' problem might arise when we have to divide some resource in a continuous manner at each stage.

The final main property we have to consider is the number of stages in the process. In the simplest case the number of stages is finite and known beforehand. There are problems, however, in which the number of stages is infinite. Such problems can arise either because we have a process in which decisions are taken sequentially in time, and we wish to continue the process indefinitely, or because decisions are being taken continuously over a given range or period, and the stages become infinitesimally small. Finally there are certain problems in which the number of stages is not known beforehand, but has to be determined in the solution.

We have, therefore, three main properties in the problems we can tackle using dynamic programming: whether the process is deterministic or not, the number of decisions at each stage in the process, and the number of stages. Fortunately, however, the formulation of problems by dynamic programming is very similar whatever the characteristics of the problem.

Two Typical Problems

To complete this outline of the field of application of the technique, let us briefly consider some typical problems.

The simplest type is one in which we have a deterministic process, with a given number of stages, and a limited number of decisions at each stage. An example of this type might be a cargo loading problem, in which we have a variety of cargoes, with different weights per unit, and different values per unit weight, and we wish to load a certain capacity with a cargo of maximum value. In this problem time is not explicit, but we can consider it as a multi-stage process, where each stage constitutes the decision on how many of each type of cargo to load. The problem is clearly deterministic in that the values and weights are

known exactly, the number of stages is known, and the number of possible decisions at each stage, though not known, is at least made finite by the capacity available.

A more complex problem is the following one of replacement. Suppose we have a set of machines with variable lives, but each life expectancy is known as a distribution of times from previous history. In any month, say, we can either replace a machine at some cost, or let it continue in service, with the chance of incurring substantial costs if it breaks down. We want to know what replacement policy leads to minimum cost. Such a problem is essentially stochastic, in that machine life is variable. The number of possible decisions at each stage is finite (in fact only two for a single machine—to replace it or not), but the number of stages is infinite if we wish to continue in the same line of business.

The Principle of Optimality

The way in which we tackle problems of this type is by applying, by sometimes devious means it is true, a very simple concept. The concept was stated by Bellman as the Principle of Optimality, in the following terms: 'An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.' This might seem rather formidable, but in essence it is equivalent to saying that if we are pursuing an optimal policy, then at each stage during the process the remaining decisions must themselves form an optimal policy from the state reached to the end-point of the process.

This is, of course, almost a truism and a proof by contradiction is immediate. For all its simplicity, however, it is very powerful in providing a means of tackling what are otherwise intractable or very difficult problems. For it enables us to relate the solution of a problem of four stages, say, to a similar problem containing five stages, and in fact to relate an n-stage problem to a $(n+1)$-stage problem. In this way we can establish a series of recurrence relations; once we can find a suitable starting-point (say the one-stage problem), we can extend this solution step by step to problems involving any number of stages.

A Simple Application of the Principle of Optimality

Before we tackle a problem in some detail, let us look briefly at a simple outline of the procedure.

Suppose we wish to travel from a town A to a town Z, and in doing so we have to pass through a series of intermediate towns, (B) to (Y). One town in each of the sets (B) to (Y) must be included, but we have a choice of routes through these. We want to find the minimum-cost route from A to Z, knowing the route costs between all towns.

The way in which we can use the Principle of Optimality in this sort of problem is as follows:

Consider first all the towns in group (B). These have associated with them costs from A, say $\alpha(B_i)$ for all i, and $\alpha(B_i)$ is the minimum (in fact here the only) cost of reaching B_i from A.

Now consider any town (C). The cost of reaching C_i depends on the intermediate town in (B) visited, but for each C_i there will be at least one of (B) giving a minimum-cost route to C_i. Thus for each C_i we have to scan all (B) to find the one giving minimum cost, say $\alpha(C_i)$. This is all straightforward so far, but when we come to the next stage, we find the minimum cost of reaching D_i by scanning all the (C) to find the one which has minimum sum $\alpha(C_j) + \text{cost } (C_j, D_i)$, i.e. the minimum-cost route to D_i includes as part of it the minimum-cost route to C_j, and we have only to scan towns (C), and not concern ourselves explicitly with towns (B). Thus we obtain the minimum-cost routes to (D) from the minimum-cost routes to (C) and have only one variable [which of the towns (C) to include] at this stage.

This procedure can be followed through until the last stage is reached, when finally the minimum-cost route from A to Z is obtained by scanning only towns (Y) to find which of these has the minimum sum $\alpha(Y_j) + \text{cost } (Y_j, Z)$. Every time we include an extra step in the process we are applying the Principle of Optimality.

Thus we are able to proceed logically step by step through the problem, and although with only four towns at each stage, and all towns in successive stages connected, there is a very large number of possible routes from A to Z, we can find the minimum-cost route by setting up 25 successive functions, each having a value at only four points and very easily computed.

This is the sort of procedure we follow in all dynamic programming problems. Let us now move on to a more practical problem in which we shall set up the type of function used in dynamic programming and indicate the computational steps required.

A Multi-Activity Investment Problem

Suppose we have a sum of money to be invested in various activities, with the relation between investment and the return known for each activity—i.e. activity i has a return $f_i(x)$ from an investment x.

We require to know how to allocate the total sum available, S, among the activities so as to obtain the maximum total return.

If there are n activities, we wish to determine that set of (x_i), where i goes from 1 to n, so that $\Sigma x_i = S$, and $\Sigma f_i(x_i)$ is maximised.

For certain types of functions, there are simple ways of solving this problem, but there are forms for $f_i(x)$ which will involve using successive approximation methods to obtain the solution. However, if n is large, such methods are very laborious, and can often be dangerous in arriving at local, but not absolute, maxima.

Dynamic programming can be used whatever the form of the functions, and though there is still a certain amount of computation required, it is much less than will usually be required in an n-variable

iteration, and there is no danger of finishing at a local maximum.

We can tackle the problem by defining a function $R_i(x)$ as being the maximum return obtainable from an investment sum x, when we allow investment only in activities 1 to i inclusive. Thus $R_1(x) = f_1(x)$ as long as $f_1(x)$ is an increasing function.

Consider now investment in activities 1 and 2; if $r_2(x, y_2)$ is the return from a total amount x when y_2 is invested in activity 2, we have

$$r_2(x, y_2) = f_1(x - y_2) + f_2(y_2)$$

But if we wish to maximise the return from our x, we must choose y_2, for the given x, so that the joint return is maximised, i.e.

$$R_2(x) = \text{Max}[r_2(x, y_2)]$$

and since y_2 may have any value between 0 and x, and since from the Principle of Optimality we must also use the amount invested in activity 1 in the optimal fashion, we have

$$R_2(x) = \underset{0 \leqslant y_2 \leqslant x}{\text{Max}} [R_1(x - y_2) + f_2(y_2)]$$

For each value of x we can find a value y_2, dependent on x, which maximises this expression. The maximum may, if the form of $f_1(x)$ is suitable, be found by differentiation, but if we do use calculus, we must ensure that we obtain the *absolute* maximum, rather than some turning value which is exceeded at other points in the range.

Of course, if we use numerical methods to obtain the maximum, we can handle any functions $f_i(x)$. In any event, we are able to compute $R_2(x)$ for all x in the range $(0, S)$, and this function gives us the maximum return obtainable from a sum x, when we permit investment in activities 1 and 2 only. (The order in which the activities are taken is unimportant, but it simplifies the notation if we take them in numerical sequence.)

The way in which the sum x is divided between the two activities depends of course on the particular functions f_1 and f_2, and for some forms the maximum return will be obtained if the whole sum x is put into one or other of the activities. The function $R_2(x)$ does not explicitly reveal how the sum is divided, but only gives the return from making the division in the best fashion. We shall, however, need to know how to make this division optimally in order to know the best policy to adopt. The information is already at hand, of course, since we have determined, for each value of x, that part y_2 which is put into activity 2 in order to obtain the maximum return.

If we denote $y_2 = y_2(x)$ as the value of y_2 which gives the maximum return from a total x, then we can regard $y_2(x)$ as a decision function, telling us how much we should invest in activity 2 when x is available for the activities 1 and 2 jointly. We now have all the information we require for the 2-stage process, and can use this to extend the solution to the 3-stage process, and sequentially to an n-stage process.

For if we now permit investment in activity 3, investing y_3 in this from a total x available, we have the return as

$$r_3(x, y_3) = f_3(y_3) + \text{return from } (x - y_3) \text{ in activities 1 and 2.}$$

But if we wish to maximise the total return, again using the Principle of Optimality, we must utilise the $(x - y_3)$ in an optimal fashion with respect to activities 1 and 2; therefore

$$r_3(x, y_3) = f_3(y_3) + R_2(x - y_3)$$

since we defined $R_2(x - y_3)$ as being the maximum return obtainable from a sum $(x - y_3)$ with investment in activities 1 and 2.

We still have to ensure that the y_3 is chosen optimally, and we do this as before, by letting it take all possible values (i.e. in the range 0, x) and choosing that particular y_3 which maximises the return, i.e. we compute

$$R_3(x) = \underset{0 \leqslant y_3 \leqslant x}{\text{Max}} [f_3(y_3) + R_2(x - y_3)]$$

Again this is easily obtained, being a single variable maximisation. This gives us the maximum return from three activities, and we may again compute this for all x in the range $(0, S)$. At the same time we automatically generate the corresponding decision function $y_3(x)$, denoting the amount y_3 which is invested in activity 3 when x is available for activities 1, 2, and 3.

We can proceed in this manner, since for all $i \geq 2$ we have

$$R_i(x) = \underset{0 \leqslant y_i \leqslant x}{\text{Max}} [R_{i-1}(x - y_i) + f_i(y_i)]$$

as the return function and $y_i = y_i(x)$ as the corresponding decision function.

Finally we have

$$R_n(x) = \underset{0 \leqslant y_n \leqslant x}{\text{Max}} [R_{n-1}(x - y_n) + f_n(y_n)]$$

for all x in $(0, S)$ as the maximum return obtainable from all n investments, and in particular

$$R_n(S) = \underset{0 \leqslant y_n \leqslant S}{\text{Max}} [R_{n-1}(S - y_n) + f_n(y_n)]$$

as the maximum return with our particular sum S available.

This tells us part of the answer, but to determine how this maximum return is obtained we have to trace back through the problem using the successive decision functions in the reverse order.

For $y_n(x)$ tells us how much we should invest in activity n, when we have x available for all activities, so that $x_n = y_n(S)$ is how much should be invested in the nth activity for our particular problem. We should note that it is only at this final stage that we commit ourselves to any particular policy, and it is this which ensures that we obtain the absolute maximum return, rather than some local maximum.

Having obtained x_n, we have $(S - x_n)$ left to invest in the remaining $(n-1)$ activities, and the function $y_{n-1}(x)$ tells us that we put $x_{n-1} = y_{n-1}(S - x_n)$ into the $(n-1)$th activity.

Similarly $x_{n-2} = y_{n-2}(S - x_n - x_{n-1})$ and we proceed back through the problem to find how much to invest in each activity.

This concludes the computation required in the problem, and though we still have an appreciable amount of work, we have reduced the labour significantly by replacing the original n-variable problem by n simple one-variable maximisations.

The Field of Solutions

With regard to the investment problem discussed in the previous section, our analysis provided far more information than the original problem required. We have, in fact, the solutions to a whole set of problems similar in form to the original, but with different parameters.

For our functions $R_n(x)$ give us the maximum return from all sums from 0 to S, and our intermediate functions $R_i(x)$, the maximum return from activities 1 to i inclusive from sums of 0 to S. This additional by-product information is typical of dynamic programming, for inherent in the application of this technique is the necessity to embed the problem in a whole field of similarly structured problems.

In an industrial environment we often require solutions to a field of problems, in order to determine what policy should be followed in a range of circumstances.

Sometimes the need to generate solutions in this way can avoid a lot of the detailed computation by giving an idea of the form of the solution. It is often possible, for instance, having defined the series of functions of which one relates to the specific problem under consideration, to see from the first few stages exactly what is the structure of the optimal policy and to infer the form of the final solution.

This procedure is helped in some problems by our having two parallel sets of functions—the return function and the decision function, for we can use either of these to help us to get an insight into the real structure of the problem.