SOME ASPECTS OF THE THEORY OF DYNAMIC PROGRAMMING

By

Richard Bellman

P-696

23 June 1955

The RAND Corporation

1700 MAIN ST. • SANTA MONICA • CALIFORNIA

Summary

The purpose of this ~~expository~~ paper is to furnish a simple introduction to the use of the theory of dynamic programming in treating multi-stage decision processes.

Some Aspects of the Theory of Dynamic Programming

By

Richard Bellman

§1. Introduction.

In recent years, a number of mathematical problems of novel and unconventional type have arisen from the study of economic, engineering, industrial, and military fields to challenge the mathematician. A particular class of these problems are those we may call "decision processes". These involve the planning, scheduling, or programming, all equivalent terms, of sequences of operations.

Many new techniques have been devised to solve these problems, and the very concept of a solution has been altered as a consequence of the availability of modern computing machines. The purpose of this paper is to present some of the basic concepts of one approach to these problems, the theory of dynamic programming.

We shall illustrate this approach by considering two simple examples, one a maximization problem of conventional type, and the other a decision process involving random or chance events. After discussing both of these problems, we shall attempt to synthesize our results and abstract the common elements.

§2. The Arithmetic Mean—Geometric Mean Inequality.

Probably the most well-proved inequality in analysis

is the arithmetic–geometric mean inequality which asserts that

$$(1) \qquad \sum_{i=1}^{n} x_i/n \geq \sqrt[n]{x_1 x_2 \ldots, x_n},$$

for any n non–negative quantities $x_1, x_2, \ldots, x_n$.

An equivalent form of the inequality is

$$(2) \qquad \max_{R} \; x_1 x_2 \ldots x_n = (1/n)^n,$$

where R is the region defined by

$$(3) \qquad \text{a.} \quad x_i \geq 0, \; i = 1,2,\ldots,n,$$

$$\text{b.} \quad \sum_{i=1}^{n} x_i = 1.$$

One may ask: What does an n–dimensional maximization problem have to do with programming a sequence of operations? The answer resides in the fact that the choice of a point in n–dimensions $(x_1, x_2, \ldots, x_n)$ may be considered to be a single operation, namely the choice of one point, or as a sequence of operations, requiring first the choice of x_1, then the choice of x_2, and so on. It is clear that there should be some analytic and computational advantages derived from replacing an n–dimensional operation by a sequence of n one–dimensional operations.

We begin with the observation that the maximum of $x_1 x_2 \ldots x_n$ over the region R is a numerical quantity depending only upon n, the dimension. Let us then define

$$(4) \qquad u_n = \max_{R} \; x_1 x_2 \ldots x_n$$

for $n = 1,2,3,\ldots$ Clearly $u_1 = 1$.

If we choose x_1 as fixed, and consider what the other x_i must be, we see that to achieve the maximum value, u_n, we must choose $x_2, x_3, \ldots, x_n$ so as to maximize the remaining product $x_2 x_3 \ldots x_n$ subject to the constraints

(5) a. $x_i \geq 0$, $i=2,\ldots,n$

b. $x_2+x_3+\ldots=1-x_1$.

Now this is a problem quite similar to the original, except for two facts. The dimension is now $(n-1)$ and the sum in (5b) is $1-x_1$, in place of 1, as in (3b).

Let us then generalize our original problem by considering the problem of maximizing $P(x) = x_1 x_2 \ldots x_n$ subject to the constraints

(6) R: a. $x_i \geq 0$

b. $\sum_{i=1}^{n} x_i = a > 0$.

The maximum of $P(x)$ will now be a function of n and a. Define the new function

(7) $f_n(a) = \operatorname{Max}_R P(x)$,

for $n = 1,2,\ldots$

For any choice of x_1 in the range $0 \leq x_1 \leq a$, we see that $x_2, x_3, \ldots, x_n$ must be chosen so as to maximize $x_2 x_3 \ldots x_n$ subject to the constraints

(8) a. $x_i \geq 0$, $i=2,3,\ldots,n$

b. $x_2+x_3+\ldots+x_n = a-x_1$.

It is clear that $x_1 \neq 0$ or a. By definition of the functions $\{f_n(a)\}$, we have

(9) $x_2x_3\ldots x_n = f_{n-1}(a-x_1)$.

Thus, for any choice of x_1 in $[\,0,a\,]$, we have

(10) $x_1x_2\ldots x_n = x_1f_{n-1}(a-x_1)$,

if we are trying to maximize. Since x_1 itself is also to be chosen to maximize the final result, we obtain the basic recurrence relation

(11) $$f_n(a) = \max_{0\leq x_1\leq a} [x_1f_{n-1}(a-x_1)].$$

For practical purposes, the problem would now be solved, since we have reduced the determination of $f_n(a)$ to the computation of a sequence of functions of one variable, namely

(12) $$f_2(a) = \max_{0\leq x_1\leq a} [x_1(a-x_1)],$$

$$f_3(a) = \max_{0\leq x_1\leq a} [x_1f_2(a-x_1)],$$

and so on.

If we insist upon the luxury of an explicit solution we may proceed as follows. From the homogeneity of all the relations, we see that

(13) $$f_n(a) = a^n f_n(1) = a^n u_n.$$

Hence (11) becomes

(14) $$a^n u_n = \max_{0 \le x_1 \le a} x_1(a-x_1)^{n-1} u_{n-1},$$

or

(15) $$u_n = \left[\max_{0 \le y \le 1} y(1-y)^{n-1} \right] u_{n-1}.$$

$$= \frac{(n-1)^{n-1}}{n^n} \quad u_{n-1}$$

Starting with $u_1 = 1$, we obtain (2) inductively.

§3. A Gold-Mining Problem.

Let us consider a problem which can more legitimately be considered a programming problem.

Suppose that we have two gold mines, Anaconda and Bonanza, and a rather delicate gold-mining machine. The properties of the machine are such that if used to mine either Anaconda or Bonanza it will bring to the surface a certain fraction of the gold in the mine, and remain undamaged, awaiting further use, or it will be irretrievably damaged and mine no gold thereafter. More precisely, we assume that if the machine is used in Anaconda, there is a probability P_1 that a fraction r_1 of the gold there will be mined and the machine remain undamaged, and a probability Q_1 that the machine will mine nothing

and be damaged beyond repair. Similarly, if Bonanza is mined, the probabilities are P_2 and Q_2 respectively, and the fraction is r_2.

The process is now the following. At the first stage we choose to use our machine in either Anaconda or Bonanza. If the machine is undamaged, we make a similar choice at the second stage, and so on, until the machine is damaged, at which time the process terminates. Given the initial quantities of gold in the mines, say x in Anaconda and y in Bonanza, the problem is to choose the sequence of operations which maximizes the expected amount of gold that is mined before the machine is defunct.

It is clear that we cannot speak of maximizing the amount of gold mined because of the probabilistic nature of the process, but that we must content ourselves with some average measure of the return.

Let us begin by observing that the expected return from an optimal sequence of choices depends only upon x and y, the initial quantities in each mine. With this in mind, we define

(1) $f(x,y)$ = expected return obtained using an optimal sequence of choices when Anaconda has x and Bonanza has y

We shall solve our problem by obtaining a recurrence relation for $f(x,y)$. This we do in the following way. Suppose we choose to mine Anaconda. If the machine is undamaged, we

obtain r_1x and are left in a situation where Anaconda possesses $(1-r_1)x$ and Bonanza possesses y. In this situation we will proceed to make an optimal sequence of choices, and hence, by definition, obtain a further expected return of $f((1-r_1)x,y)$. Since the probability that the machine is undamaged is P_1, we see that the expected return from an initial choice of Anaconda is

$$(2) \qquad f_A(x,y) = P_1(r_1x+f((1-r_1)x,y)).$$

Similarly, the expected return from an initial choice of Bonanza is

$$(3) \qquad f_B(x,y) = P_2(r_2y+f(x,(1-r_2)y)).$$

Since we want to choose the mine which maximizes the total return, the final equation for $f(x,y)$ is

$$(r) \qquad f(x,y) = \text{Max}\ (f_A(x,y),\ f_B(x,y))$$

$$= \text{Max} \begin{bmatrix} \text{A:} & P_1(r_1x+f((1-r_1)x,y)), \\ \text{B:} & P_2(r_2y+f(x,(1-r_2)y)) \end{bmatrix}$$

We have thus reduced the original problem to the analytic problem of solving this unconventional functional equation. In this case also, we can solve explicitly. It can be shown that the rule which determines the choice of Anaconda or Bonanza is the following:

$$\text{a. If } \frac{P_1 r_1 x}{1-r_1} > \frac{P_2 r_2 y}{1-r_2}, \text{ choose Anaconda,}$$

$$\text{(5) b. If } \frac{P_1 r_1 x}{1-r_1} < \frac{P_2 r_2 y}{1-r_2}, \text{ choose Bonanza,}$$

$$\text{c. If } \frac{P_1 r_1 x}{1-r_1} = \frac{P_2 r_2 y}{1-r_2}, \text{ choose either.}$$

Observe that what we call the "solution", is not an analytic expression for $f(x,y)$, which is relatively unimportant, but a rule for carrying out the optimal sequence of choices, which is, after all, what we wanted. The functional equation merely serves as an intermediary for the determination of the optimal sequence of choices.

§4. Analysis.

Let us now see if we can extract the common features of these quite dissimilar problems. In so doing we shall be able to recognize other problems which may be treated by the same techniques.

In each problem the status of the process is described by a small number of parameters. In the maximization problem these were the number of variables remaining to be chosen, and their sum a; in the gold-mining problem these were the quantities of gold available in the two mines at the beginning of any stage. Furthermore, at each stage of either

process, we had a choice of decisions. In the maximization problem, we had a choice of x_1 between 0 and a; in the gold-mining problem, we had a choice of a mine essentially a choice of A or B. The effect of these decisions was to transform the descriptive parameters into another similar set. In the maximization problem, we reduce it to $a-x_1$ by a choice of x_1 and reduce the number of stages remaining; in the gold-mining problem, we alter the amount of gold in one mine or the other, if the machine is undamaged.

The essential feature of these problems is a certain invariance, in the sense that at each stage we are confronted by a situation of the same general type. Only processes possessing this symmetry over time can be treated by the techniques of the theory of dynamic programming. For processes escaping these methods, there remain the computational algorithms of linear and nonlinear programming, Monte Carlo techniques, and, occasionally, for want of better, the brute force of computing machines.

§5. Abstraction.

We can describe this invariance in the following way. We have an abstract system, S, whose state is characterized at any time by the vector $P = (P_1, P_2, \ldots, P_n)$. At each stage of the process, we have a choice of a number of decisions, or transformations, which convert P into a new vector $T(P,q)$, where q symbolizes the decision we make. The purpose of the multi-stage process is to maximize some prescribed function

of the final state, with the problem that of determining the sequence of decisions which produces the maximum.

Let us call any sequence of allowable decisions a policy, and a policy that yields the maximum an optimal policy.

The fundamental aim of the theory is the determination of the structure of all optimal policies. In order to obtain an analytic hold on the problem we introduce the return function $f(P)$, the total return obtained using an optimal policy starting from the state P.

The recurrence relations obtained in the previous sections (3.4) and (2.11), are both consequences of the following intuitive and plausible principle:

Principle of Optimality. An optimal policy has the property that whatever the first decision, the remaining decisions must constitute an optimal policy for the state resulting from the first decision.

Using this principle, the functional equation governing the general process described above is

$$(1) \qquad f(P) = \operatorname{Max}_{q} f(T(P,q)).$$

Applications of this technique have been made to various parts of mathematical economics, to the theory of control processes, and to such fields of mathematics as the calculus of variations and the theory of differential equations.

Those interested in further details may consult the following two articles:

1. "The Theory of Dynamic Programming", Bull. Amer. Math. Soc., Vol. 60 (1954), pp. 503-516.

2. "Dynamic Programming—A Survey", Jour. Operations Research Society, Vol. 2 (1954), pp. 275-289.

where many further references will be found.