Introduction to Dynamic Programming

Leon Cooper and Mary W Cooper
Southern Methodist University, Dallas, Texas

International Series in Modern Applied Mathematics and Computer Science Volume 1

Pergamon Press

Introduction to Dynamic Programming

by

LEON COOPER

and

MARY W. COOPER

Southern Methodist University, Dallas, Texas, USA

PERGAMON PRESS

OXFORD · NEW YORK · TORONTO · SYDNEY · PARIS · FRANKFURT

U.K.	Pergamon Press Ltd., Headington Hill Hall, Oxford OX3 0BW, England
U.S.A.	Pergamon Press Inc., Maxwell House, Fairview Park, Elmsford, New York 10523, U.S.A.
CANADA	Pergamon of Canada, Suite 104, 150 Consumers Road, Willowdale, Ontario M2J 1P9, Canada
AUSTRALIA	Pergamon Press (Aust.) Pty. Ltd., P.O. Box 544, Potts Point, N.S.W. 2011, Australia
FRANCE	Pergamon Press SARL, 24 rue des Ecoles, 75240 Paris, Cedex 05, France
FEDERAL REPUBLIC OF GERMANY	Pergamon Press GmbH, 6242 Kronberg-Taunus, Hammerweg 6, Federal Republic of Germany

First edition 1981

British Library Cataloguing in Publication Data

Cooper, Leon

Introduction to dynamic programming. — (Pergamon international library: international series in modern applied mathematics and computer science; vol. 1).

1. Dynamic programming

I. Title

519.7'03 T57.83 79-42640

ISBN 0-08-025065-3 Hardcover

ISBN 0-08-025064-5 Flexicover

Printed in Hungary by Franklin Printing House

Preface

THE purpose of this book is to present a clear and reasonably self-contained introduction to dynamic programming. We have tried to steer a middle ground between presentation of the underlying mathematical ideas and results, and the application of these ideas to various problem areas. Consequently, there is a large number of solved practical problems as well as a large number of computational examples to clarify the way dynamic programming is used to solve problems.

This book is definitely introductory and makes no claim to cover the whole subject; if an attempt had been made to approach completeness the book would be many times larger than it is. However, We hope that it will give a good insight into the fundamental ideas of the subject, a good working knowledge of the relevant techniques and an adequate starting-point for further study into those regions of the subject not dealt with here.

A consistent notation has been applied throughout for the expression of such quantities as state variables, decision variables, etc. It is felt that this should be helpful to the reader in a field where the notational conventions are far from uniform. In Section 6.9 a new application is presented and in Section 7.7 a new method for state dimensionality is presented.

The book should be suitable for self-study or for use as a text in a one-semester course on dynamic programming at the senior or first-year, graduate level for students of mathematics, statistics, operations research, economics, business, industrial engineering, or other engineering fields.

We are indebted to the many comments of students in a course on dynamic programming who used this book in the form of preliminary lecture notes.

Southern Methodist University
Dallas

LEON COOPER
MARY W. COOPER

Contents

Chapter 1

Introduction

1.1. Optimization

Dynamic programming is a particular approach to optimization. By optimization what we usually mean is finding the best solution to some problem from a set of alternatives. In this book we shall consider "problems" which can be quantitatively formulated. Hence we shall deal with *mathematical models* of situations or phenomena which exist in the real world. By a mathematical model, we mean that we have abstracted or isolated from an incomparably richer background, certain salient features which can be described in the language of mathematics. If we have a valid isolate, i.e., the characteristics that we have ignored have negligible effects, then we can expect that the solution to our model will provide a deeper understanding and a reasonably accurate description of the phenomenon under study. The generation of suitable models in the sense just described is more of an art than an exact science. Nevertheless, it is a widely practiced art and one which is increasingly successful. The development of powerful computational tools, i.e., digital computers, has had a very strong impact on the impetus to develop increasingly complex and sophisticated mathematical models.

Let us now consider the basic components of any mathematical optimization model. These are:

1. *Variables* (or decision variables or policy variables or independent variables). These are the quantities or factors that can be manipulated to achieve some desired outcome or objective. Most often variables will be represented as $x_1, x_2, \ldots, x_n$ or the vector of variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. We will often speak of $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as a point in an n-dimensional Euclidean space (see Appendix).

2. *Objective function* (or return function or profit function). This is a measure of the effectiveness or the value or utility which is associated with some particular combination of the variables. In many instances it is some single-valued function of the variables, i.e., $z = f(x_1, x_2, \ldots, x_n)$. This is the function that is to be optimized (maximized or minimized). In most instances, the function $z = f(\mathbf{x})$ is a known function. However, there are optimization problems in which this is not the case. In these problems, usually considered in the calculus of variations (and also in a different way in dynamic programming), the function itself is not known and is the solution to the optimization problem. As an example of such problems, we might minimize some integral I, where

$$I = \int_a^b F(x, y, y')\, dx \tag{1.1.1}$$

In (1.1.1) $y = f(x)$ is the particular function (and is to be determined) that will minimize I and $F(x, y, y')$ is some known function of x, y, and $y' \equiv dy/dx$. Hence there are two distinct kinds of objective functions. In the first kind we seek values of $\mathbf{x} = (x_1 \ldots x_n)$ that maximizes or minimizes $f(\mathbf{x})$. In the second kind, we seek, as our unknown, a function $y = f(x)$ that maximizes or minimizes a *functional*, $I = \int_a^b F(x, y, y')\,dx$. There are more complex examples of this type which will be dealt with later.

3. *Constraints* (or feasibility conditions). These are algebraic equations or inequalities or in some cases differential equations, that the variables must satisfy, in addition to providing a maximum or minimum value of the objective function. Usually constraints can be represented as

$$\left.\begin{aligned} h_i(x_1, x_2, \ldots, x_n) &\leqslant 0 \\ h_i(x_1, x_2, \ldots, x_n) &= 0 \\ h_i(x_1, x_2, \ldots, x_n) &\geqslant 0 \end{aligned}\right\} \quad (i = 1, 2, \ldots, m) \tag{1.1.2}$$

where for a given i, only one of the three may hold. In some instances, notably in variational problems, differential equation constraints may also be present. For example, we might wish to minimize

$$I(y) = \int_a^b F(x, y, y')\,dx$$

subject to

$$\frac{dy}{dx} = h(x, y) \tag{1.1.3}$$

$$y(a) = C$$

In summary, the general variable optimization problem is to

maximize[†]

$$y = f(x)$$

subject to

$$h_i(\mathbf{x})\{\leqslant, =, \geqslant\}0 \qquad (i = 1, 2, \ldots, m) \tag{1.1.4}$$

We shall not consider general variational problems at this point. Details of the application of dynamic programming to variational problems will be presented in a later chapter.

1.2. Separable Functions

An important criterion which is often used to characterize functions appearing in either the objective function or the constraints of a mathematical optimization (or *mathematical programming*, as it is usually called) problem is whether or not the functions are *separable*. A separable function is one in which the function consists of a sum of functions of a single variable, i.e.,

$$f(\mathbf{x}) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n) \tag{1.2.1}$$

[†] It should be apparent that $\min[f(\mathbf{x})] = -\max[-f(\mathbf{x})]$. Hence we can restrict our concerns to either maximization or minimization problems.

There are other forms of "separability" which will be dealt with in subsequent chapters. The form shown in (1.2.1) is probably the most often used in dynamic programming problem formulations.

At first glance, it would seem that the form shown in (1.2.1) restricts one unduly, since many important functions are not separable. For example,

$$f(x_1, x_2) = 3x_1x_2+2x_1 \sin x_2$$

is clearly *not* in separable form. However, it is not difficult to see that most (but not all) functions can be transformed into separable form by introducing auxiliary variables and additional constraints. In [2] it is shown that the method for separating any "factorable" function (defined below) consists of two basic steps which are repeatedly used until one obtains a separable function. These two steps are:

(1) Replace any product term of the form

$$h_1(x_1)\,h_2(x_2) \quad \text{by} \quad y_1^2-y_2^2$$

and add the constraints

$$h_1(x_1) = y_1-y_2, \quad h_2(x_2) = y_1+y_2$$

(2) Replace any term of the form

$$H(h(x)) \quad \text{by} \quad H(y)$$

and add the constraint

$$h(x) = y$$

The class of nonlinear functions which can be separated by repeated use of the above steps comprises what have been called *factorable* functions. A *factorable function* is a function of n variables which is generated by first composing (adding or multiplying) functions of a single variable, transforming those functions, composing those, ..., etc., a finite number of times.

As an example of how this process operates, suppose we wished to put the following problem in separable form:

$$\underset{\{x_1, x_2\}}{\text{maximize}} \quad x_1e^{(x_1+x_2)^2} \tag{1.2.2}$$

Applying step (1) we have

$$\begin{aligned} &\max_{\{x_1, x_2, y_1, y_2\}} && y_1^2-y_2^2 \\ \text{subject to} \quad &&& y_1-y_2 = x_1 \\ &&& y_1+y_2 = e^{(x_1+x_2)^2} \end{aligned} \tag{1.2.3}$$

The second constraint in (1.2.3) is still not in separable form so we now apply step (2) to obtain

$$\begin{aligned} &\max_{\{x_1, x_2, y_1, y_2, y_3\}} && y_1^2-y_2^2 \\ \text{subject to} \quad &&& y_1-y_2 = x_1 \\ &&& y_1+y_2 = e^{y_3} \\ &&& (x_1+x_2)^2 = y_3 \end{aligned} \tag{1.2.4}$$

The last constraint in (1.2.4) is nonseparable so we reapply step (2) to obtain

$$\max_{\{x_1, x_2, y_1, y_2, y_3, y_4\}} \quad y_1^2 - y_2^2$$

subject to

$$\begin{aligned} y_1 - y_2 &= x_1 \\ y_1 + y_2 &= e^{y_3} \\ y_4^2 &= y_3 \\ x_1 + x_2 &= y_4 \end{aligned} \tag{1.2.5}$$

Equation (1.2.5) is now a mathematical programming problem in completely separable form and is completely equivalent to (1.2.2).

1.3. Convex and Concave Functions

There is an important property of certain functions which is related to the existence of minima and maxima. This property is known as *convexity* and its opposite as *concavity*. In Fig. 1.3.1(a) we have shown a convex function of a single variable. Intuitively,

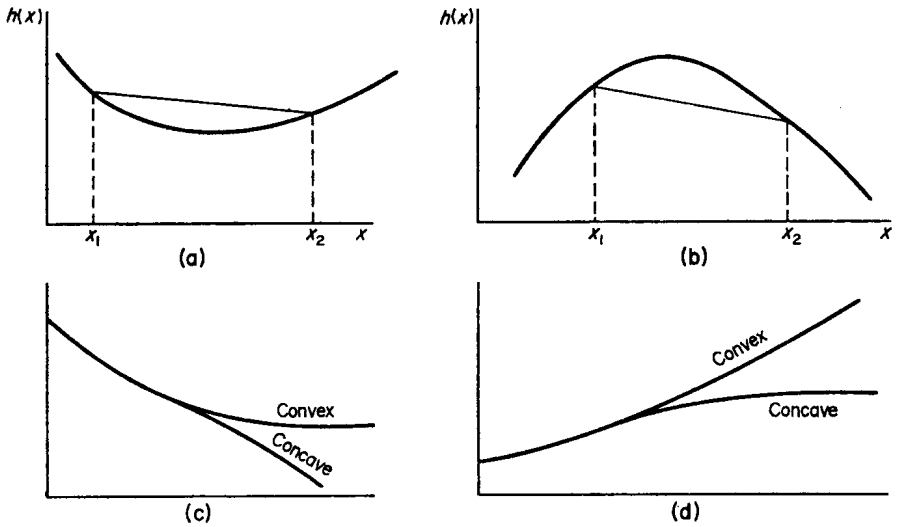

FIG. 1.3.1. Convex and concave functions of a single variable.

as Fig. 1.3.1(a) shows, a function of a single variable is *convex* if a line segment drawn between any two points on its graph falls entirely on or above the graph. Similarly, a function is *concave* if a line segment drawn between any two points on its graph falls entirely on or below the graph. Such a function is shown in Fig. 1.3.1(b). If the line segment falls entirely above (below) the graph the function is said to be *strictly convex (strictly concave)*.

Formally, a function $f(x)$ of a single variable x is said to be convex over some interval in x, if for any two points x_1, x_2 in the interval and for all λ, $0 \leqslant \lambda \leqslant 1$,

$$f[\lambda x_1 + (1-\lambda)x_2] \leqslant \lambda f(x_1) + (1-\lambda) f(x_2) \tag{1.3.1}$$

Similarly, a function $f(x)$ is said to be concave over some interval in x, if for any two points x_1, x_2 in the interval and for all λ, $0 \leqslant \lambda \leqslant 1$,

$$f[\lambda x_1+(1-\lambda)x_2] \geqslant \lambda f(x_1)+(1-\lambda) f(x_2) \tag{1.3.2}$$

It should be clear from these definitions that if $f(x)$ is convex, $-f(x)$ is concave and vice versa. It will be seen that (1.3.1) and (1.3.2) correspond to the definition given in terms of the graph of a function.

We can generalize the definitions of convexity and concavity just given to the case of functions of several variables. A function $f(x)$ is *convex* over some convex set† X in E^n if for any two points $\mathbf{x}_1$ and $\mathbf{x}_2$ in X and for all λ, $0 \leqslant \lambda \leqslant 1$,

$$f[\lambda \mathbf{x}_1+(1-\lambda)\mathbf{x}_2] \leqslant \lambda f(\mathbf{x}_1)+(1-\lambda) f(\mathbf{x}_2) \tag{1.3.3}$$

Similarly, a function $f(x)$ is *concave* over some convex set X in E^n if for any two points $\mathbf{x}_1$ and $\mathbf{x}_2$ in X and for all λ, $0 \leqslant \lambda \leqslant 1$,

$$f[\lambda \mathbf{x}_1+(1-\lambda)\mathbf{x}_2] \geqslant \lambda f(\mathbf{x}_1)+(1-\lambda) f(\mathbf{x}_2) \tag{1.3.4}$$

For a function of a single variable, a function was convex if the line segment joining any two points on its curve fell entirely on or above the curve. A similar interpretation can be given to (1.3.3). A function $z = f(\mathbf{x})$ is a hypersurface in $(n+1)$-dimensional space. It is convex if the line segment which connects any two points $(\mathbf{x}_1, z_1)$ and $(\mathbf{x}_2, z_2)$ on the surface of this hypersurface lies entirely on or above the hypersurface.

Some properties relating to convex and concave functions that we shall have occasion to refer to in subsequent chapters are as follows. Proofs of these assertions can be found in Cooper and Steinberg [1].

PROPOSITION 1.3.1. *If the functions $h_k(\mathbf{x})$, $k = 1, 2, \ldots, s$ are convex functions over some convex set X in E^n, then the function $h(\mathbf{x}) = \sum_{k=1}^{s} h_k(\mathbf{x})$ is also a convex function over X.*

What Proposition 1.3.1 asserts is that the sum of convex functions is also a convex function over some convex set. For a function of a single variable, it is clearly true. The convex set is the x axis (or some portion of it) and, as an example, we show in Fig. 1.3.2 the sum of two convex functions $h_1(x) = x^2$ and $h_2(x) = 2x$ over $0 \leqslant x \leqslant 4$.

Just as the sum of convex functions is a convex function, it is also true that the sum of concave functions over a convex set is a concave function. Hence, this leads us to:

PROPOSITION 1.3.2. *If the functions $h_k(\mathbf{x})$, $k = 1, 2, \ldots, s$ are concave functions over some convex set X in E^n, then the function $h(\mathbf{x}) = \sum_{k=1}^{s} h_k(\mathbf{x})$ is also a concave function.*

A result of some importance is the following:

PROPOSITION 1.3.3. *If $h(x)$ is a convex function over the nonnegative orthant of E^n, then if $W = \{\mathbf{x} \mid h(\mathbf{x}) \leqslant b, \mathbf{x} \geqslant 0\}$ is not empty, W is a convex set.*

† See the Appendix.

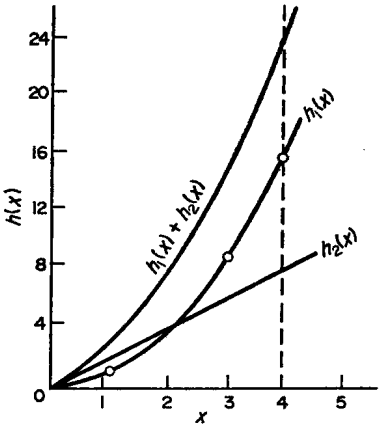

FIG. 1.3.2. Sum of convex functions.

A simple example of Proportion 1.3.3 is shown in Fig. 1.3.3. We have chosen $h(\mathbf{x}) = 2x_1^2+3x_2^2$. It is easily verified that this is a convex function. If we examine the set $W = \{\mathbf{x} \mid h(\mathbf{x}) \leqslant 12, \mathbf{x} \geqslant 0\}$, we see that it is a convex set.

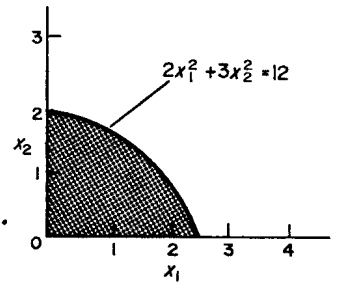

FIG. 1.3.3. $h(x) \leqslant 12$.

In a similar fashion, we can show the following:

PROPOSITION 1.3.4. *If $h(x)$ is a concave function over the nonnegative orthant of E^n, then if $W = \{\mathbf{x} \mid h(\mathbf{x}) \leqslant b, \mathbf{x} \geqslant 0\}$ is not empty, W is a convex set.*

1.4. Optima of Convex and Concave Functions

Generally, if we are seeking the maximum or minimum of some arbitrary function $z = f(\mathbf{x})$ and we have no particular characterization of the function, we can only hope to find a local maximum or minimum with any existing methods. This is true even if we know as much about the function as that it is both continuous and differentiable in the region of interest. However, if the function to be maximized or minimized over some convex region is either convex or concave, then there is considerably more information available to us. In addition, great simplifications in computational procedures can often be made. To this end we shall state two results of great importance. Proofs of these propositions can be found in [1].

PROPOSITION 1.4.1. *Let $h(\mathbf{x})$ be a convex function over a closed convex set X in E^n. Then any local minimum of $h(\mathbf{x})$ in X is also the global minimum of $h(\mathbf{x})$ in X.*

The importance of Proposition 1.4.1 is very great. It gives us a condition under which we do not need to be concerned about the presence of many local maxima or minima.

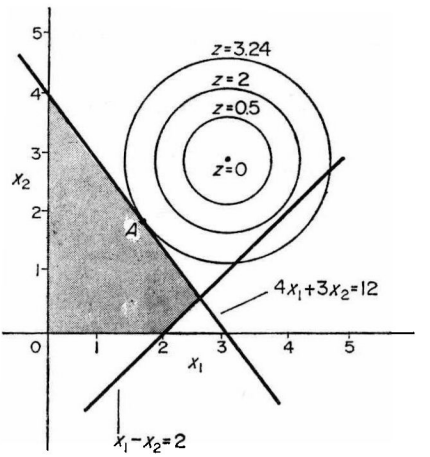

FIG. 1.4.1. Optimal solution on constraint boundary.

An example illustrating this result is shown graphically in Fig. 1.4.1. The function $z = (x_1-3)^2+(x_2-3)^2$ is a convex function. The constraints

$$\begin{aligned} 4x_1+3x_2 &\leqslant 12 \\ x_1-x_2 &\leqslant 2 \\ x_1 &\geqslant 0 \\ x_2 &\geqslant 0 \end{aligned} \tag{1.4.1}$$

generate a convex set as is seen in Fig. 1.41. Hence, we know immediately that the solution we have obtained, i.e., $(x_1, x_2) = (1.56, 1.92)$ and $z = 3.24$, is a globally minimum solution. In this case it is also unique.

In a similar fashion one can show a corresponding result for concave functions.

PROPOSITION 1.4.2. *Let $h(\mathbf{x})$ be a concave function over a closed convex set X in E^n. Then any local maximum of $h(\mathbf{x})$ in X is also the global maximum of $h(\mathbf{x})$ in X.*

Propositions 1.4.1 and 1.4.2 deal with the case of minimizing a convex function or maximizing a concave function over a convex set. Let us now consider the reverse case, i.e., maximizing a convex function or minimizing a concave function over a convex set. This gives rise to the following results.

PROPOSITION 1.4.3. *Let X be a closed convex set bounded from below and let $h(\mathbf{x})$ be a convex function over X in E^n. If $h(\mathbf{x})$ has global maxima at points x^0 with $|x^0|$ finite, then one or more of the $\mathbf{x}^0$ are extreme points of X.*

PROPOSITION 1.4.4. *Let X be a closed convex set bounded from below and let $h(\mathbf{x})$ be a concave function over X in E^n. If $h(\mathbf{x})$ has global minima at points $\mathbf{x}^0$ with $|\mathbf{x}^0|$ finite, then one or more of the $\mathbf{x}^0$ are extreme points of X.*

What these results tell us is that we can be sure that the global maximum or minimum will be at an extreme point of the convex set of feasible solutions. Since the number of extreme points is often finite, this often results in computationally feasible methods for examining some subset of this finite set.

An example of Proposition 1.4.4 is the following problem:

$$\begin{aligned} \max \quad & z = x_1^2 + x_2^2 \\ & -x_1 + x_2 \leqslant 4 \\ & x_1 + x_2 \leqslant 12 \\ & x_1, x_2 \geqslant 0 \end{aligned} \tag{1.4.2}$$

The problem of (1.4.2) is shown in Fig. 1.4.2. We can see that the global maximum occurs at an extreme point, designated A, of the convex set generated by the constraints of (1.4.2). At this point the maximum value $z = 144$ is obtained.

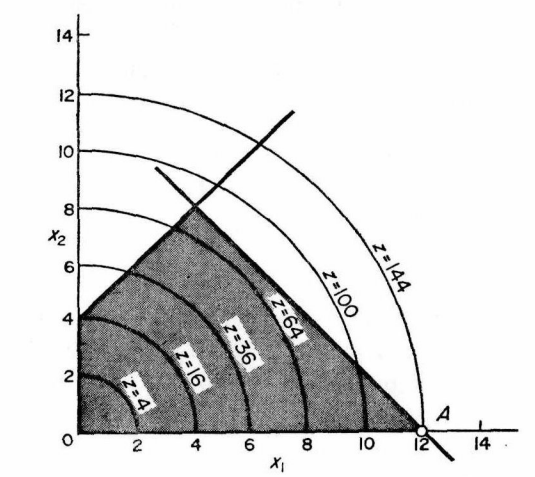

FIG. 1.4.2. Extreme point solution.

1.5. Dynamic Programming

Dynamic programming is a particular "approach" to optimization. We use the word "approach" advisedly because dynamic programming is not a particular algorithm in the sense that Euclid's algorithm is a well-defined procedure for finding the greatest common divisor of two integers or in the sense that Dantzig's simplex algorithm is a well-defined set of rules for solving a linear programming problem. Dynamic programming is an approach to solving certain kinds of optimization problems, some of which

can, in principle, be solved by other procedures. Dynamic programming is a way of looking at a problem which may contain a large number of interrelated decision variables so that the problem is regarded as if it consisted of a sequence of problems, each of which required the determination of only one (or a few) variables. Ideally, what we seek to do is, in effect, substitute solving n single variable problems for solving one n variable problem. Whenever this is possible, it usually requires very much less computational effort. Solving n smaller problems requires a computational effort which is proportional to n, the number of single variable problems if each problem contains one variable. On the other hand, solving one larger problem with n variables usually requires a computational effort which is very roughly proportional to a^n, where a is some constant. Hence the desirability of transforming or considering an n-dimensional problem as n one-dimensional problems.

The principle or point of view that enables us to carry out the transformation we have just discussed is known as the *principle of optimality*. It was first enunciated by Bellman [3]. It has an intuitively obvious basis (Bellman's justification consists of the single statement, "A proof by contradiction is immediate"). We shall have more to say about this later. The principle of optimality is:

An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with respect to the state which results from the initial decision.

A simpler rendition of this principle, which shows its intuitive character more transparently, consists of the statement: *Every optimal policy consists only of optimal subpolicies.*

It is important, even if only vaguely and intuitively at this point, to understand what the above statements of the principle of optimality mean. Consider the following optimization problem. An investor has a fixed sum of money, D dollars, which can be invested in five different investment opportunities (stocks, bonds, land, etc.). In actual fact he wishes to invest this money *at the same time* (the present), assuming that he has a prediction of what kind of return he can expect from each investment. Since each investment has certain stipulations that must be met (minimum or maximum amounts required, deferred earnings, different tax rates, etc.), he wants to know how much of his total D should he invest in each of the five investments so as to maximize the total return from all the investments.

One approach to solving such a problem is to generate all possible combinations of the five investments and see which one provides the largest return. This approach is called *total enumeration* and is generally not practical, even with computers, for problems with a realistic number of variables. If we assume that the amount to be invested in any of the investment opportunities is fixed, the decision that must be made is to invest or not invest. Even for this simplified case we have one way to invest in all of the investment opportunities, five ways to invest in four of them, ten ways to invest in three of them, ten ways to invest in two of them, and five ways to invest in one, i.e., the total number of ways to invest in five investment opportunities (for a fixed investment amount in each one) is

$$\binom{5}{5}+\binom{5}{4}+\binom{5}{3}+\binom{5}{2}+\binom{5}{1}=31$$

For 20 investments we would have to evaluate 1,048,575 combinations. The number increases very rapidly with the number of investments, and hence total enumeration rapidly becomes impractical, even on a computer.

What we shall do in dynamic programming is examine only a very small subset of the total number of combinations. However, the subset is guaranteed, under the right conditions, to contain the optimal solution. Hence we shall find this optimal solution. Let us now return to our five-investment problem and consider the dynamic programming approach. Even though all investments are to be made at one time, we shall pretend that the investment decisions are made in some order. What this means is that we invest some amount I_1 in investment 1, amount I_2 in investment 2, etc. The only restriction that we have is that the sum of $I_1+I_2+I_3+I_4+I_5 = D$ and that the investments are nonnegative amounts. Now let us apply the principle of optimality. What it says is that no matter how much of the total amount D has been spent on the first k investments, the remaining money must be optimally distributed among the remaining 5-k investments. Let us now apply this principle.

Suppose we had numbered the investments 1 through 5 as indicated above. If we have already spent a certain amount of the money on the first four investments, then whatever remains will be invested in the fifth investment opportunity. Hence no decision is made here. Let us now consider how much should be invested in I_4. Here we make a decision but it is a very simple decision. It involves deciding how much to invest in I_4 and how much to save for I_5. This involves a decision involving only *one variable*, i.e., how much to invest in I_4. Let us now consider how much to invest in I_3. Again we are involved in a single variable decision process, i.e., how much to invest in I_3 and how much to save for the combined investments in I_4 and I_5. This analysis continues for I_2 and I_1. In brief, the decision maker has to make five simple one-variable decisions instead of one complex five-variable decision. What we are saying when we make a decision is, e.g., that when we decide how much to invest in I_3, no matter how much we invest in I_1 and I_2, the amount we invest in I_3 must be optimal with respect to the remaining capital. Since we do not yet know what we shall invest in I_4 and I_5, the optimal investment and return from I_3 must be determined for *all* feasible amounts remaining to be invested. The details of how this is done will be discussed in the next chapter.

1.6. Dynamic Programming: Advantages and Limitations

We have already alluded to one of the main advantages of dynamic programming. That is, that when we use dynamic programming, we transform a single n-dimensional optimization problem into n one-dimensional optimization problems which can be solved one at a time. The classical extremum methods of analysis cannot do this.

A second extremely important advantage of dynamic programming over almost all other extant computational methods, and especially classical optimization methods, is that dynamic programming determines absolute (global) maxima or minima rather than relative (local) optima. Hence we need not concern ourselves with the vexing problem of local maxima and minima.