OPERATIONS RESEARCH

Chapter 8

Dynamic Programming

Prof. Bibhas C. Giri

Department of Mathematics

Jadavpur University

Kolkata, India

Email: bcgiri.jumath@gmail.com

MODULE - 1: Basic Concept and Terminology, and Dynamic Programming Models I and II

1.0 Introduction

Consider a company that has to decide on the production plan of an item for the next three months so as to meet the demands in different months at minimum cost. The different months for which the production is to be decided constitute the stages. So it is a *multi-stage problem*. For such a problem, decisions are made sequentially over several periods using a technique called *Dynamic Programming (DP)* technique. One thing common to all models in this category is that current decisions influence both present and future periods.

The dynamic programming approach divides the problem into several sub- problems or stages and then these sub-problems are solved sequentially until the initial problem is finally solved. The common characteristic of all dynamic programming models is expressing the decision problem by means of recursive formulation.

1.1 Terminology

Terminologies which are commonly used in dynamic programming are given below:
Stage: The point at which a decision is made is known as a stage. The end of a stage marks the beginning of the immediate succeeding stage. For instance, in the salesmen allocation problem, each territory represents a stage; in the shortest route problem, each city represents a stage.

State: The variable that links two stages in a multistage decision problem is called a state variable. At any stage, the values that state variables can take describe the status of the problem. These values are referred to as states. For example, in the shortest route problem, a city is referred to as state variable.

Principle of optimality: The principle of optimality states that the optimal decision from any state in a stage to the end, is independent of how one actually arrives at that state.

Optimal policy: A policy which optimizes the value of an objective function is called an optimal policy.

Bellman's principle of optimality : It states that "an optimal policy (a sequence of decisions) has the property that whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Return function : At each stage, a decision is made which can affect the state of the system at the next stage and help in arriving at the optimal solution at the current stage. Every decision has its merit which can be represented in an algebraic equation form. This equation is called a *return function*.

1.2 Characteristics of Dynamic Programming

The basic features which characterize the dynamic programming problem are as follows:

(a) The problem can be subdivided into stages with a policy decision required at each stage. A stage is a device to sequence the decisions. That is, it decomposes a problem into sub-problems such that an optimal solution to the problem can be obtained from the optimal solutions to the sub-problems.

(b) Every stage consists of a number of states associated with it.

(c) Decision at each stage converts the current stage into state associated with the next stage.

(d) The state of the system at a stage is described by a set of variables, called *state variables*.

(e) When the current state is known, an optimal policy for the remaining stages is independent of the policy of the previous ones.

(f) To identify the optimal policy for each state of the system, a recursive equation is formulated with n stages remaining, given the optimal policy for each state with $(n-1)$ stages left.

(g) Using recursive equation approach each time, the solution procedure moves backward stage by stage for obtaining the optimum policy of each state for that particular stage, till it attains the optimum policy beginning at the initial stage.

1.3 Dynamic Programming Algorithm

The computational procedure for solving a problem by dynamic programming approach can be summarized in the following steps:

Step 1. Decompose (or divide) the given problem into a number of smaller sub-problems (or stages). Identify the state variables at each stage and write down the transformation function as a function of the state variables and decision variables at the next stage.

Step 2. Write down a general recursive relationship for computing the optimal policy. Decide whether forward or backward method is to be followed to solve the problem.

Step 3. Construct appropriate stages to show the required values of the return function at each stage.

Step 4. Determine the overall optimal policy or decisions and its value at each stage. There may be more than one such optimal policy.

Note:

1. *Generally the solution of a recursive equation involves two types of computations, according as the system is continuous or discrete. In the first case, the optimal decision at each stage is obtained by using the usual classical methods of optimization. In second case, a tabular computational scheme is followed.*

2. *If the dynamic programming problem is solved by using the recursive equation starting from the first through the last stage, i.e., obtained the sequence $f_1 \to f_2 \to ... \to f_N$, the computation involved is called the forward computational procedure. If the recursive equation is formulated in a different way so as to obtain the sequence $f_N \to f_{N-1} \to ... \to f_1$ then the computation is known as the backward computational procedure.*

1.4 Dynamic Programming Model I

Single additive constraint and multiplicatively separable return

Consider the problem:

$$\text{Maximize} \quad z = \prod_{j=1}^{n} f_j(y_j)$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_j y_j = b, \;\; y_j \geq 0, \; a_j \geq 0.$$

We introduce the state variables $s_j = \sum a_j y_j = b$, $s_{j-1} = s_j - a_j y_j$, $j = 2, 3, \cdots, n$.

Let $F_j(s_j) = \max_{y_1, y_2, \cdots, y_j} \prod_{1}^{j} f_j(y_j)$. Then the general recursion formula is given by

$F_j(s_j) = \max_{y_j} \left[f_j(y_j) F_{j-1}(s_{j-1}) \right]$, $j = n, n-1, \cdots, 2$ and $F_1(s_1) = f_1(y_1)$.

Example 1.1: Maximize $y_1 y_2 y_3$ subject to $y_1 + y_2 + y_3 = 5$; $y_1, y_2, y_3 \geq 0$

Solution: We introduce the state variables $s_3 = y_1 + y_2 + y_3$, $s_2 = s_3 - y_3 = y_1 + y_2$, $s_1 = s_2 - y_2 = y_1$. Let $F_j(s_j) = \max_{y_1, y_2, \cdots y_j} \prod_{1}^{j} y_j$. Then

$F_3(s_3) = \max_{y_3} [y_3 F_2(s_2)]$

$F_2(s_2) = \max_{y_2} [y_2 F_1(s_1)]$

$F_1(s_1) = s_2 - y_2$.

Hence, $F_2(s_2) = \max_{y_2}[y_2(s_2 - y_2)]$.

We use differential calculus to maximize $y_2(s_2 - y_2)$ and get the optimal value of y_2 as $y_2 = s_2/2$.

Then, using Bellman's principle of optimality, we have

$$F_3(s_3) = \max_{y_3}[y_3 s_2^2/4] = \max_{y_3}[y_3(s_3 - y_3)^2/4]$$

Using differential calculus again, we find that $y_3 = s_3/3 = 5/3$ maximizes $y_3(s_3 - y_3)^2/4$. This gives $s_2 = y_1 + y_2 = 10/3$. Thus we obtain $y_2 = s_2/2 = 5/3$, and $y_1 = 5/3$. Hence the maximum value of $y_1 y_2 y_3$ is 125/27.

1.5 Dynamic Programming Model II

Single additive constraint and additively separable return

Consider the problem in which the objective or return function z is an additively separable function of n variables y_j and $f_j(y_j)$ is a function of y_j. Find y_j, $1 \le j \le n$, which minimize $z = \sum_{j=1}^{n} f_j(y_j)$, subject to the constraint $\sum_{j=1}^{n} a_j y_j \ge b$, where $a_j (\ge 0)$ and $b (> 0)$ are real numbers, $y_j \ge 0$, $j = 1, 2, \cdots, n$.

This is an n-stage problem where suffix j indicates the stage. Since values of y_j's are to be decided, y_j's are called decision variables. The return at the jth stage is the function $f_j(y_j)$. Thus, each decision y_j is associated with a return $f_j(y_j)$.

We introduce state variables $s_1, s_2, \cdots, s_n$ such that

$$\begin{aligned}
s_n &= a_1 y_1 + a_2 y_2 + \cdots + a_n y_n \ge b \\
s_{n-1} &= a_1 y_1 + a_2 y_2 + \cdots + a_{n-1} y_{n-1} = s_n - a_n y_n \\
s_{n-2} &= a_1 y_1 + a_2 y_2 + \cdots + a_{n-2} y_{n-2} = s_{n-1} - a_{n-1} y_{n-1} \\
\cdots &\quad \cdots\cdots \cdots \cdots \cdots \cdots \cdots \\
s_1 &= a_1 y_1 = s_2 - a_2 y_2
\end{aligned}$$

Let $s_{j-1} = T_j(s_j, y_j)$, $2 \le j \le n$, where T_j is the *stage transformation function*.

Let $F_n(s_n)$ denote the minimum value of z for any feasible value of s_n, where s_n being the function of all decision variables. Then

$$F_n(s_n) = \min_{y_1, y_2, \cdots, y_n} [f_1(y_1) + f_2(y_2) + \cdots + f_n(y_n)], \quad s_n \ge b.$$

If we choose a particular value of y_n and minimize z over the remaining $n-1$ variables then we have

$$F_n(s_n) = f_n(y_n) + \min_{y_1, y_2, \cdots, y_{n-1}} \left[\sum_{j=1}^{n-1} f_j(y_j) \right] = f_n(y_n) + F_{n-1}(s_{n-1})$$

Therefore, the minimum over all y_n for any feasible s_n is given by

$$F_n(s_n) = \min_{y_n} \; [f_n(y_n) + F_{n-1}(s_{n-1})].$$

If the value of $F_{n-1}(s_{n-1})$ is known for all y_n, then the function which is to be minimized involves only a single variable y_n. The recursion formula is

$$\begin{aligned} F_j(s_j) &= \min_{y_j}[f_j(y_j) + F_{j-1}(s_{j-1})], \quad 2 \le j \le n \\ \text{and } F_1(s_1) &= f_1(y_1). \end{aligned}$$

Now, starting with $F_1(s_1)$ we recursively optimize to obtain $F_2(s_2), F_3(s_3), \cdots$ and finally we obtain $F_n(s_n)$ for feasible s_n. Each time optimization is done over a single variable only.

Example 1.2: Minimize $z = y_1^2 + y_2^2 + y_3^2$, subject to $y_1 + y_2 + y_3 \ge 15$, and $y_1, y_2, y_3 \ge 0$.

Solution: Here the decision variables are y_1, y_2 and y_3. We define the state variables s_1, s_2 and s_3 such that

$$\begin{aligned} s_3 &= y_1 + y_2 + y_3 \ge 15 \\ s_2 &= y_1 + y_2 = s_3 - y_3 \\ s_1 &= y_1 = s_2 - y_2 \end{aligned}$$

Also, we define $F_j(s_j) = \min_{y_j}[f_j(y_j) + F_{j-1}(s_{j-1})]$ for $j = 2, 3$ and $F_1(s_1) = f_1(y_1)$. Then we have

$$\begin{aligned} F_3(s_3) &= \min_{y_3}[y_3^2 + F_2(s_2)] \\ F_2(s_2) &= \min_{y_3}[y_2^2 + F_1(s_1)] \\ F_1(s_1) &= y_1^2 = (s_2 - y_2)^2 \\ \text{Thus we have } F_2(s_2) &= \min_{y_2}[y_2^2 + (s_2 - y_2)^2] \end{aligned}$$

Using calculus, we see that $y_2^2 + (s_2 - y_2)^2$ is minimum for $y_2 = s_2/2$. Hence $F_2(s_2) = s_2^2/2$.
Now, $F_3(s_3) = \min_{y_3}[y_3^2 + F_2(s_2)] = \min_{y_3}[y_3^2 + (s_3 - y_3)^2/2]$, using Bellman's principle.
Again, using calculus, we see that $y_3^2 + (s_3 - y_3)^2/2$ is minimum for $y_3 = s_3/3$.

Hence, $F_3(s_3) = s_3^2/3,\ s_3 \geq 15$.
Since $F_3(s_3)$ is minimum for $s_3 = 15$, therefore, the minimum value of $y_1^2 + y_2^2 + y_3^2$ is 75, where $y_1 = y_2 = y_3 = 5$.

Example 1.3: Use Dynamic Programming to show that

$$z = p_1 \log p_1 + p_2 \log p_2 + ... + p_n \log p_n$$
$$\text{subject to } \ p_1 + p_2 + ... + p_n = 1 \text{ and } p_j \geq 0,\ j = 1, 2, \cdots, n$$

is a minimum when $p_1 = p_2 = ... = p_n = 1/n$.

Solution: The problem here is to divide unity into n parts so as to minimize the quantity $\sum_i p_i \log p_i$.
Let $f_n(1)$ denote the minimum attainable sum $\sum_i p_i \log p_i$.
For $n = 1$ (stage 1), we have

$$f_1(1) = p_1 \log p_1 = 1 \log 1 \ \text{ as } p_1 = 1.$$

For $n = 2$, the unity is divided into two parts p_1 and p_2, such that $p_1 + p_2 = 1$.
If $p_1 = x$ and $p_2 = 1 - x$, then we have

$$\begin{aligned} f_2(1) &= \min\{p_1 \log p_1 + p_2 \log p_2\} \\ &= \min_{0<x\leq 1} \{x \log x + (1-x)\log(1-x)\} \\ &= \min_{0<x\leq 1} \{x \log x + f_1(1-x)\} \end{aligned}$$

In general, for an n-stage problem, the recursive equation is

$$\begin{aligned} f_n(1) &= \min\{p_1 \log p_1 + p_2 \log p_2 + ... + p_n \log p_n\} \\ &= \min_{0<x\leq 1} \{x \log x + f_{n-1}(1-x)\} \end{aligned}$$

We now solve this recursive equation.
For $n = 2$ (stage 2), the function $x \log x + (1-x)\log(1-x)$ attains its minimum value at $x = 1/2$ satisfying the condition $0 < x \leq 1$. Thus,

$$f_2(1) = \frac{1}{2}\log\frac{1}{2} + (1 - \frac{1}{2})\log(1 - \frac{1}{2}) = 2(\frac{1}{2}\log\frac{1}{2}).$$

Similarly, for stage 3, we have

$$\begin{aligned} f_3(1) &= \min_{0<x\leq 1} \{x \log x + f_2(1-x)\} \\ &= \min_{0<x\leq 1} \{x \log x + 2(\frac{1-x}{2})\log(\frac{1-x}{2})\} \end{aligned}$$

Now, since the minimum value of $x\log x + 2(\frac{1-x}{2})\log(\frac{1-x}{2})$ is attained at $x = 1/3$ satisfying $0 < x \leq 1$, we have

$$f_3(1) = \frac{1}{3}\log\frac{1}{3} + 2(\frac{1}{3}\log\frac{1}{3}) = 3(\frac{1}{3}\log\frac{1}{3}).$$

Therefore, the optimal policy is $p_1 = p_2 = p_3 = 1/3$.

In general, for n-stage problem, we assume that the optimal policy is

$$p_1 = p_2 = ... = p_n = 1/n \text{ and } f_n^0(1) = n\left\{\frac{1}{n}\log\frac{1}{n}\right\}.$$

This can be shown easily by mathematical induction. For $n = m+1$, the recursive equation is

$$\begin{aligned} f_{m+1}(1) &= \min_{0<x\leq 1}\{x\log x + f_m(1-x)\} \\ &= \min_{0<x\leq 1}\left[x\log x + m\left\{\frac{1-x}{m}\log\left(\frac{1-x}{m}\right)\right\}\right] \\ &= \frac{1}{m+1}\log\frac{1}{m+1} + m\left\{\frac{1}{m+1}\log\frac{1}{m+1}\right\} \\ &= (m+1)\left\{\frac{1}{m+1}\log\frac{1}{m+1}\right\}, \end{aligned}$$

since minimum of $x\log x + m(\frac{1-x}{m}\log\frac{1-x}{m})$ is attained at $x = \frac{1}{m+1}$.

Hence, the required optimal policy is

$$\left(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right) \text{ with } f_n^0(1) = n\left(\frac{1}{n}\log\frac{1}{n}\right).$$