NOTES ON THE THEORY OF
DYNAMIC PROGRAMMING—VI
THE WAREHOUSING MODEL

By

Richard Bellman

P-773

29 November 1955

Approved for OTS release
Summary

The purpose of this paper is to show how the functional equation technique of the theory of dynamic programming yields a very simple computational algorithm for the solution of mathematical models arising in stock level studies.

A numerical solution of these problems relying upon linear programming techniques had previously been given by Charnes and Cooper.
NOTES ON THE THEORY OF DYNAMIC PROGRAMMING—VI
THE WAREHOUSING MODEL

By
Richard Bellman

§1. Introduction

In a recent report, [2], Charnes and Cooper present a solution by means of linear programming techniques of one version of what is called the "warehouse problem". As formulated by A. Cahn, [1], it reads

"Given a warehouse with fixed capacity and an initial stock of a certain product, which is subject to known seasonal price and cost variations, what is the optimal pattern of purchasing (or production), storage and sales?"

The purpose of this note is to indicate how problems of this general nature may be approached by means of the functional equation technique of the theory of dynamic programming, and thereby reduced to a very simple and straightforward computational problem.

In order to compare the two approaches more readily, we shall use the formulation and notation of Charnes and Cooper, [2].

§2. Analytic Formulation

Following the report of Charnes and Cooper, let
B = the fixed warehouse capacity
A = the initial stock in the warehouse

Consider a seasonal product to be bought (or produced) and sold for each of \( i = 1, 2, \ldots, n \) periods. For the \( i^{th} \) period, let

\begin{align*}
(1) & \quad c_i = \text{cost per unit} \\
(2) & \quad p_i = \text{selling price per unit} \\
(3) & \quad x_i = \text{amount bought (or produced)} \\
(4) & \quad y_i = \text{amount sold}
\end{align*}

The constraints are as follows:

\begin{enumerate}
\item \textbf{Buying constraint:} The stock on hand at the end of the \( i^{th} \) period cannot exceed the warehouse capacity.
\item \textbf{Selling constraint:} The amount sold in the \( i^{th} \) period cannot exceed the amount available at the end of the \((i - 1)^{st}\) period.
\item \textbf{Non-negativity constraint:} Amounts purchased or sold in any period are non-negative.
\end{enumerate}

Analytically:

\begin{align*}
(3a) & \quad A + \sum_{j=1}^{i} (x_j - y_j) \leq B, \quad i = 1, 2, \ldots, n, \\
(3b) & \quad y_i \leq A + \sum_{j=1}^{i-1} (x_j - y_j), \quad i = 1, 2, \ldots, n, \\
(3c) & \quad x_i, y_i \geq 0.
\end{align*}
The problem is to determine the quantities $x_i$ and $y_i$ so as to maximize the over-all profit

$$P = \sum_{j=1}^{n} (p_j y_j - c_j x_j).$$

§3. Dynamic Programming Treatment

It is clear that the maximum profit will be a function of the original quantity of stock, $A$, and the duration of the process $n$. Define

$$f_n(A) = \max P,$$

where the maximum is taken over all admissible values of the $x_i$ and $y_i$. We have

$$f_1(A) = \max (p_1 y_1 - c_1 x_1),$$

over all $x_1, y_1$ satisfying

$$y_1 \leq A$$

$$A + (x_1 - y_1) \leq B,$$

or

$$f_1(A) = p_1 A.$$

We now wish to derive a recurrence relation connecting $f_n(A)$ and $f_{n+1}(A)$. If $x_i, y_i$ are chosen, the constraints on the remaining variables are
(5) 
(a) \[ \sum_{j=2}^{1} (x_j - y_j) \leq B - (A + (x_i - y_i)), \]
(b) \[ y_1 \leq (A + (x_1 - y_1)) + \sum_{j=2}^{1-1} (x_j - y_j). \]

Hence, for \( n \geq 2 \),

(6) \[ f_n(A) = \max_{x_1, y_1} \left[ \sum_{i=1}^{n} (x_i y_i - c_i x_i) + f_{n-1} (A + x_i - y_i) \right], \]

where the maximum is taken over the region

(7) 
(a) \[ y_1 \leq A \]
(b) \[ x_1 - y_1 \leq B - A, x_1, y_1 \geq 0. \]

The variable \( A \) assumes all values in the interval \([0, B]\).

§4. Discussion

Let us now discuss the actual computation of the solution. As far as the memory and tabulation problems are concerned, we are dealing with a sequence of functions of one variable. Consequently, no difficulties arise from this direction.

The maximization, however, is over a two dimensional region, and a variable region at that. Hence, we might expect that the computation would be slowed down by this fact. Fortunately, we are rescued by the linearity of the process.

Consider the region defined by the equations in (3.7)

\[ x_1 \]
\[ x_1 - y_1 = B - A \]
We suspect that the maximum will occur at one of the vertices, and this may be established rigorously in several ways, either directly from linear programming, or in an inductive fashion. In the figure above, we have assumed that $A > B - A$ or $2A > B$. In this case, the vertices are

(1) $P_1(0, 0), P_2(0, B - A), P_3(A, 0), P_4(A, B)$.

If $A < B - A$, there are only three vertices

(2) $P_1(0, 0), P_2(0, B - A), P_5(B - A, 0)$.

Taking all five vertices as possible maximizing points, which takes care of the two cases $B - A > A$ at one time, we can reduce (3.6) to

(3) $f_n(A) = \max \begin{bmatrix}
1. f_{n-1}(A), \\
2. -c_1(B - A) + f_{n-1}(B), \\
3. p_1A \\
4. p_1A - c_1B + f_{n-1}(B) \\
5. p_1(B - A) + f_{n-1}(2A - B)
\end{bmatrix}$

for $A \geq 0$, with $f_n(A) = 0$ for $A \leq 0$.

This computation is now a very simple one. The quantity $B$ is taken as fixed, and $A$ assumes all values in the interval $[0, B]$. 
Bibliography
