

## CYCLIC DYNAMIC PROGRAMMING: A PROCEDURE FOR PROBLEMS WITH FIXED DELAY

David G. Luenberger

*Stanford University, Stanford, California*

(Received June 15, 1970)

A dynamic decision problem in which the effect of control action is either delayed for a number of time periods or has an effect that lasts for a fixed period leads, in the conventional formulation, to a high-dimensional dynamic-programming problem. This paper develops a method for exploiting the inherent structure of such problems that leads to low-dimensional 'cyclic dynamic programming' problems that are easier to solve than the problem that is obtained by the conventional approach.

**I**N MANY dynamic allocation problems the effect of some decision may be delayed for a fixed period after the decision is made, or the effect may have only finite duration. A simple example of a problem of this type is to find the optimum building schedule of nuclear-fuel reprocessing plants, having only finite lifetime of  $L$  years, in order to meet a known reprocessing demand in each  $T$  years. Thus a sequence of building decisions is made in time, the effect of each lasting only  $L$  years into the future. Here, the finite lifetime is caused by the fact that plants have only a finite period of economical operation. Other problems of this type include determining a schedule for cutting and planting trees in a given area assuming that only trees older than  $L$  years will be cut, various equipment-replacement problems, large construction projects having essentially fixed construction times, and control problems involving communication lags.

Fixed delays or finite lifetimes often arise as a result of contracts or laws. Thus, one must deal with fixed-term insurance policies, with guarantees, with prepayment clauses in loans and bonds, and with numerous other fixed-period structures. Indeed, because they are created from language rather than from hardware or natural elements, legal documents offer a variety of examples having ideal fixed delays or finite lifetimes.

In a physical environment, the pure fixed-lifetime model is admittedly somewhat artificial in some cases. More realistic would be a probabilistic lifetime, or a lifetime that could be varied by paying an additional cost. These more complex model structures cannot, however, presently be attacked directly by existing optimization techniques. One method for attacking them is to solve a series of problems, each having a fixed-lifetime structure, and then select the best of these. The technique developed in this paper can therefore be regarded as a step toward the solution of these sometimes more realistic but exceedingly complex lifetime or delay problems.

Mathematically, problems with fixed delays or finite lifetimes are characterized by the fact that each decision affects not only the state of the system at the next

stage but also the state of the system a given number  $L$  of stages later. This means that the natural system state cannot serve as a state in the sense of dynamic programming.

The usual prescription for handling problems with delay of this type by dynamic programming is to augment the original state space by  $L$  additional variables. In particular, if  $x_t$  is the state at  $t$  and  $u_t$  denotes the decision variable at period  $t$ , then the new state vector  $X_t = (x_t, u_{t-1}, u_{t-2}, \dots, u_{t-L})$  evolves in a Markovian manner and dynamic programming can be employed.<sup>[1]</sup> This technique, even for problems with relatively short delays, increases the dimensionality of the problem beyond the reach of present-day computing capabilities.

This paper describes a method that takes advantage of the unique structure of problems with delays or finite lifetimes and essentially reduces them to equivalent dynamic programming problems of modest dimension in many cases. The number of additional variables that must be introduced by the method is equal to one less than the integer  $n$  satisfying  $(n-1)L < T \leq nL$ . Thus the method is most effective for problems in which the delay period  $L$  is a significant fraction of the planning horizon  $T$ .

The technique is demonstrated first in a simple plant-building example and then stated in general form.

### THE PLANT-BUILDING-SCHEDULE EXAMPLE

IN THIS EXAMPLE, plants with a finite lifetime of  $L$  years are to be built in order to meet a given fixed demand during a planning horizon of  $T$  years. Economies of scale favor the building of a few high-capacity plants, while the discount factor on future expenditures favors numerous low-capacity ones.

#### A. Formulation

The mathematical form of the problem is

$$\text{minimize } \sum_{t=0}^{T-1} [1/(1+a)^t]c(u_t), \quad (1)$$

subject to

$$x_{t+1} = x_t + u_t - \begin{cases} 0, & 0 \leq t \leq L-1, \\ u_{t-L}, & L \leq t \leq T-1, \end{cases} \quad (2)$$

$$x_0 = 0, \quad x_t \geq d_t, \quad 1 \leq t \leq T, \quad (3)$$

where

$T$  = planning horizon— $T = 16$  years in the example.

$L$  = plant lifetime (a plant built in year  $t$  operates in year  $t+1$  through year  $t+L$ )— $L = 10$  years in the example.

$u_t$  = capacity built in year  $t$ .

$c(u)$  = cost of building capacity  $u$ .

$a$  = interest rate.

$x_t$  = total capacity operating in year  $t$ .

$d_t$  = demand in year  $t$  ( $u_t$ ,  $x_t$ , and  $d_t$  are expressed in the same normalized unit).

In most situations, the function  $c$  will, because of economies of scale, be con-

cave increasing, which implies that the problem probably possesses numerous local minima.

If conventional dynamic programming approaches are applied to this problem, the best that can be done is to include in the state vector at time  $t$  the present total capacity and the capacities of all previously built plants that will die between times  $t+1$  and  $T$ . The (linear) dimension of this space is 7 for several stages of our example. Since, with present computing capabilities, only problems with less than four or five linear dimensions are solvable in a reasonable time, a new approach is required.

### B. Cyclic Dynamic Programming Method

The crucial observation is that what is built in a given year is relevant only the following year and 10 years later and does not affect directly the decisions made in between. Thus, for instance, knowledge of  $x_0, x_{10}$  and  $u_0, u_{10}$  determines both  $x_1$  and  $x_{11}$  uniquely. More precisely, the set of equations (2) can be replaced by the equivalent system (with stage index  $k$ )

$$\begin{cases} x_{k+1} = x_k + u_k, \\ x_{k+L+1} = x_{k+L} - u_k + u_{k+L}, \\ x_{k+1} = x_k + u_k. \end{cases} \quad \begin{array}{l} 0 \leq k \leq T-L-1=5, \\ T-L=6 \leq k \leq L-1=9. \end{array} \quad (4)$$

After introducing

$$X_k = \begin{cases} \begin{bmatrix} x_k \\ x_{k+L} \end{bmatrix}, & 0 \leq k \leq 6, \\ x_k, & 7 \leq k \leq 10, \end{cases} \quad U_k = \begin{cases} \begin{bmatrix} u_k \\ u_{k+L} \end{bmatrix}, & 0 \leq k \leq 5, \\ u_k, & 6 \leq k \leq 9, \end{cases}$$

equation (4) becomes

$$X_{k+1} = I_k X_k + A_k U_k, \quad 0 \leq k \leq 9, \quad (5)$$

with

$$I_k = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & 0 \leq k \leq 5, \\ (1 \ 0), & k=6, \\ 1, & 7 \leq k \leq 9, \end{cases} \quad (6)$$

and

$$A_k = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, & \text{for } 0 \leq k \leq 5, \\ 1, & \text{for } 6 \leq k \leq 9. \end{cases} \quad (7)$$

Defining

$$D_k = \begin{cases} \begin{bmatrix} d_k \\ d_{k+L} \end{bmatrix}, & 0 \leq k \leq 6, \\ d_k, & 7 \leq k \leq 9, \end{cases} \quad (8)$$

$$C_k(U_k) = \begin{cases} c(u_k) + [1/(1+a)^L]c(u_{k+L}), & 0 \leq k \leq 5, \\ c(u_k), & 6 \leq k \leq 9, \end{cases} \quad (9)$$

(3) and (1) can be expressed as

$$X_k \geq D_k, \quad 0 \leq k \leq 9, \quad (10)$$

$$\text{minimize } \sum_{k=0}^{k=9} [1/(1+a)^k] C_k(U_k). \quad (11)$$

Problem (11) subject to (5) and (10) is a dynamic program of only  $L=10$  stages with  $X_k$  as state vector and  $U_k$  as control vector. The state space has dimension 2 during the first 7 stages and dimension 1 during the last 3 stages, as shown in Fig. 1.

An important feature of the problem in this new form is the boundary-condition structure. We have

$$X_0 = \begin{bmatrix} x_0 \\ \vdots \\ x_{10} \end{bmatrix}, \quad X_{10} = x_{10}, \quad (12)$$

where  $x_0=0$  is given. The variable  $x_{10}$  is not specified, but it appears in both the initial and final states, and hence the two ends of the problem are tied together. The dynamic-programming procedure must account for this circular structure, where each stage is coupled to the succeeding one and the last is coupled to the first. For this reason we have termed the resulting formulation 'cyclic dynamic programming.' After generalizing the formulation we direct attention to the solution of this form of problem in the section after next.

### THE GENERAL CASE

A SEQUENTIAL decision problem of the form

$$(I) \begin{cases} \text{minimize } \sum_{t=0}^{T-1} l_t(x_t, u_t), & \text{subject to} & (13) \\ x_{t+1} = f(x_t, u_t, t), & t=0, \dots, L-1, x_0 \text{ given,} & (14) \\ x_{t+1} = f(x_t, u_t, u_{t-L}, t), & t=L, \dots, T-1, & (15) \\ g(x_t, u_t, t) \leq 0, & t=0, \dots, T, & (16) \end{cases}$$

where  $x_t$  and  $u_t$  are the state and the control variables and can be placed in the cyclic-dynamic-programming form. [At a slight increase in notational complexity, the whole discussion can be trivially extended to the case where both  $x_t$  and  $u_t$  are vectors.] First, one replaces the time horizon  $T$  by  $n$  folds of length  $L$ , where  $n$  is the integer such that

$$(n-1)L < T \leq nL. \quad (17)$$

Then, after introducing

$$X_k = \begin{bmatrix} x_k \\ x_{k+L} \\ \vdots \\ x_{k+(n-1)L} \end{bmatrix}, \quad 0 \leq k \leq T - (n-1)L, \quad (18)$$

$$X_k = \begin{bmatrix} x_k \\ x_{k+1} \\ \vdots \\ x_{k+(n-2)L} \end{bmatrix}, \quad T - (n-1)L + 1 \leq k \leq L, \quad (19)$$

$$U_k = \begin{bmatrix} u_k \\ u_{k+L} \\ \vdots \\ u_{k+(n-1)L} \end{bmatrix}, \quad 0 \leq k \leq T - (n-1)L - 1, \quad (20)$$

$$U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+(n-2)L} \end{bmatrix} \quad T - (n-1)L \leq k \leq L-1, \quad (21)$$

equations (14) and (15) can be expressed by the equivalent set

$$X_{k+1} = \varphi_k(X_k, U_k), \quad k=0, \dots, L-1, \quad (22)$$

**STAGE  $k = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10$**

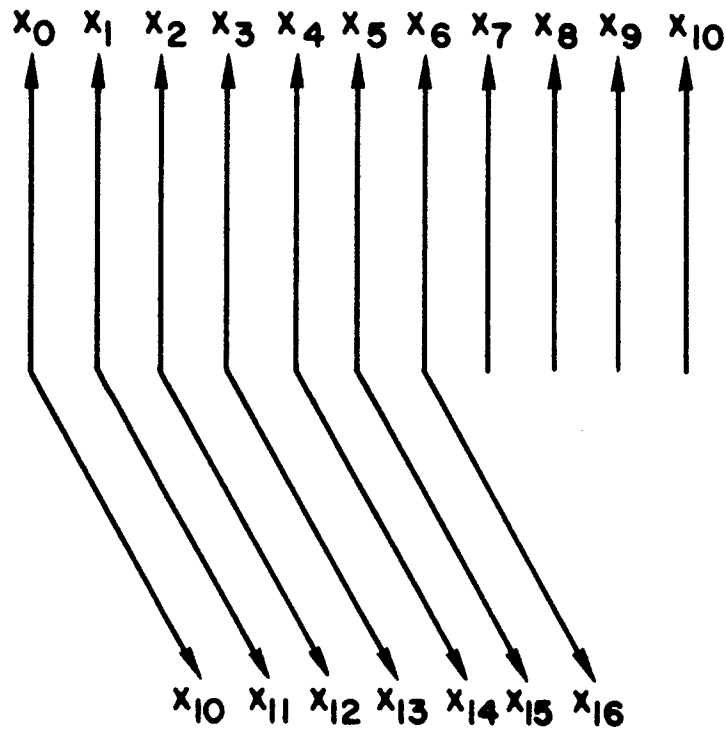


Figure 1

while the constraints of type (16) are replaced by

$$\gamma_k(X_k, U_k) \leq 0, \quad k=0, \dots, L, \quad (23)$$

and the objective function takes the form

$$\text{minimize } \sum_{k=0}^{L-1} L_k(X_k, U_k), \quad (24)$$

where

$$\begin{aligned} L_k(X_k, U_k) &= \sum_{i=0}^{n-1} l_{k+iL}(x_{k+iL}, u_{k+iL}), & 0 \leq k \leq T - (n-1)L - 1, \\ L_k(X_k, U_k) &= \sum_{i=0}^{n-2} l_{k+iL}(x_{k+iL}, u_{k+iL}), & T - (n-1)L \leq k \leq L-1. \end{aligned} \quad (25)$$

In addition, there is the consistency condition, which is reflected in the boundary condition

$$\begin{bmatrix} x_0 \\ X_L \end{bmatrix} = X_0, \quad (26)$$

which is a condition coupling the initial and final states. In the case  $T = nL$  the boundary condition is

$$\begin{bmatrix} x_0 \\ X_L \end{bmatrix} = \begin{bmatrix} X_0 \\ x_T \end{bmatrix}.$$

An interesting special case occurs when (14) and (15) are linear, i.e.,

$$x_{t+1} = A_t x_t + B_t u_t + \begin{cases} 0, & t=0, \dots, L-1, \\ C_t u_{t-L}, & t=L, \dots, T-1. \end{cases}$$

Assuming for simplicity that  $T = nL$ , so that the time horizon is integrally divided by the delay length, the linear system takes the form

$$X_{k+1} = \begin{bmatrix} A_k & & & & 0 \\ & A_{k+1} & & & \\ & & \ddots & & \\ & & & & \\ 0 & & & & A_{k+(n-1)L} \end{bmatrix} X_k + \begin{bmatrix} B_k & 0 & 0 & & 0 \\ & B_{k+1} & & 0 & \\ C_k & & \cdot & & \\ & C_{k+L} & & \cdot & 0 \\ & & \cdot & \cdot & \\ & & & & 0 \\ 0 & & C_{k+(n-1)L} & & B_{k+(n-1)L} \end{bmatrix} U_k, \quad 0 \leq k \leq L-1.$$

### SOLUTION METHODS

THE TWO-POINT boundary condition on the state vector can be treated directly by extending the dynamic programming algorithm or indirectly by using Lagrange multipliers. This section discusses both methods, although it gives primary emphasis to the Lagrange-multiplier method, since it best conserves dimensionality.

For the direct approach, we observe from (26) that, if the optimal value of  $X_L$  were known, then the general problem (I) would reduce to the  $n$ -dimensional problem (22), (23), and (24), with uncoupled boundary conditions at both ends. [ $X_L$  is generally an  $(n-1)$ -vector in the special case where  $T = nL$ ; the  $X_L$  considered in the discussion will, for simplicity, mean the first  $n-1$  components of  $X_L$ ,  $x_T$  being disregarded.] Such a problem can be solved by a standard application of either forward or backward dynamic programming. The problem can therefore be regarded as a search over the unknown vector  $X_L$ —with a dynamic-program execution being required to evaluate each search point. The storage space required for this procedure is essentially only that needed for a single dynamic program of  $n$  dimensions, but the computational time may be large.

Alternatively, the entire procedure can be carried out by a single dynamic-programming scheme, thereby conserving computing time, but at the expense of large storage requirements. At each stage  $k$  set up a copy of  $\mathfrak{X}_L \times \mathfrak{X}_k$ , the space of ordered pairs  $(X_L, X_k)$ . Then (assuming backward dynamic programming is used) the required recursion for the optimal cost  $I_k(X_L, X_k)$  from stage  $k$  to the end, terminating at  $X_L$  and starting in stage  $k$  at  $X_k$  is

$$I_k(X_L, X_k) = \min_{U_k, \gamma_k(X_k, U_k) \leq 0} \{I_{k+1}[X_L, \varphi_k(X_k, U_k)] + L_k(X_k, U_k)\}.$$

This set of computations can be carried out most efficiently in order:

Set  $I_k(X_L, X_k) = \infty$  for all  $(X_L, X_k)$ .

For each fixed  $X_k$ , vary  $U_k$  over all possibilities satisfying  $\gamma_k(X_k, U_k) \leq 0$ , calculating  $\varphi_k(X_k, U_k)$  and  $L_k(X_k, U_k)$ ; then, for each  $X_L$ , calculate

$$I_k(X_L, X_k) = \min\{I_k(X_L, X_k), I_{k+1}[X_L, \varphi_k(X_k, U_k)] + L_k(X_k, U_k)\}.$$

In this form, each  $U_k$  is used only once for each  $X_k$ , and the results used to calculate  $I_k(X_L, X_k)$  for all  $X_L$  simultaneously.

Lagrange multipliers offer perhaps the most practical method for solving the cyclic problem, for they essentially break the circular connection imposed by the constraints and allow the use of standard forward or backward dynamic programming. Actually, the method can be regarded as a special procedure for performing the search required in the direct method.

After introducing the  $(n-1)$ -dimensional vector  $\lambda$ , redefine

$$L_0(X_0, U_0) = (0, \lambda^T)X_0 + \sum_{i=0}^{n-1} l_i(x_{iL}, u_{iL})$$

and

$$L_{L-1}(X_{L-1}, U_{L-1}) = \sum_{i=0}^{n-2} l_{L-1+iL}(x_{L-1+iL}, u_{L-1+iL}) - \lambda^T \varphi_{L-1}(X_{L-1}, U_{L-1}),$$

while leaving all other quantities in (25) unchanged. Then define

$$\Gamma(\lambda) = \min \sum_{k=0}^{L-1} L_k(X_k, U_k), \quad (27)$$

subject to (22) and (23). This is a standard dynamic-programming problem obtained from the cyclic problem by deleting the constraint (26) and adjoining the term  $\lambda^T\{(0, I)X_0 - X_L\}$  to the objective functional. The resulting functional  $\Gamma$  is termed the *dual functional* for these constraints.<sup>[5]</sup>

For a given  $\lambda$  (with components  $\lambda_i$ ) the problem (27), (22), and (23) can be interpreted as an optimal planning problem similar to the original one, except that at the points  $t=iL$ ,  $i=1, 2, \dots, n-1$  there is an additional opportunity to buy or sell units of  $x_{iL}$  directly at the price  $\lambda_i$ . The resulting optimal policy will not, in general, satisfy the cyclic boundary condition—the resulting mismatches being attributed to the extra buying and selling.

Under appropriate conditions<sup>[2,3]</sup> (for example, if mismatches at  $t=iL$  are introduced in the original problem and the optimal objective value is a convex function of these mismatches, there will be an appropriate Lagrange-multiplier vector yielding perfect match), there exists a Lagrange-multiplier vector  $\lambda^0$  such that the solution to (27), (22), and (23) is identical to the solution of the cyclic problem, and, hence, perforce, to the original problem (I). Furthermore, this Lagrange multiplier maximizes the dual functional  $\Gamma$ .

There are various schemes for searching for the optimal  $\lambda^0$ . One common method, based on the economic interpretation of the Lagrange multipliers given above, is to note whether, at a given price  $\lambda_i$ , the optimal policy calls for buying or selling  $x_{iL}$ . If for buying, then the price is too low and should be increased; if for selling, then profit is being made on excess  $x_{iL}$  and the price should be reduced. In equilibrium there will be no buying or selling. Another approach is to exploit the maximizing property of  $\lambda^0$  and apply one of the variety of available unconstrained maximization techniques to  $\Gamma$ .<sup>[2,4,5]</sup>

## APPLICATION TO THE PLANT BUILDING SCHEDULE EXAMPLE

THE TECHNIQUE of cyclic dynamic programming was tested on the plant building schedule example described in the second section. In the example tested, the additional requirement that  $0 \leq u_t \leq 8$ ,  $u_t$  integer, was introduced; the interest rate was set at  $\alpha = 0.2$  and the minimal demand requirements  $d_t$  were specified as illustrated in Fig. 2.

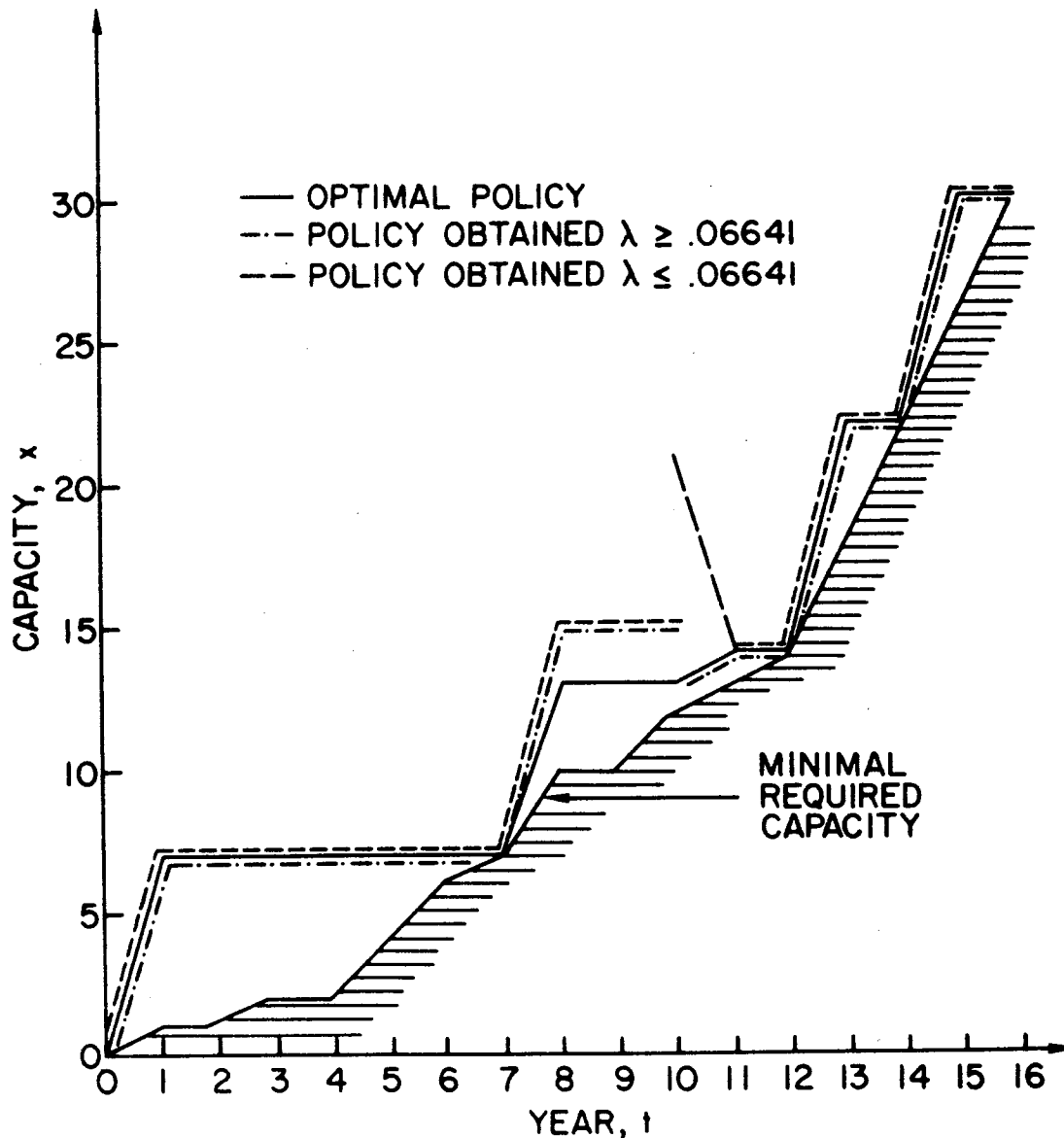


Figure 2

In the first example tested the *concave* building cost  $c(u) = u^{0.20}$  was used. The problem was put in the cyclic-dynamic-programming form and solved by forward dynamic programming for each possible  $x_{10}$ . We recall that  $x_{10}$  enters both initial and final conditions. The total search took less than two and a half minutes of IBM 360/65 and the optimum solution was found to be that shown in Table I.

As can be expected from the concavity of the objective function, no Lagrange

TABLE I  
SOLUTION TO THE CONCAVE CASE

$t$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$u$	7	0	0	0	0	0	0	6	0	0	8	0	8	0	8	0

Total cost: 3.3423.

multiplier exists for this problem and hence the special procedure described in the preceding section cannot yield the optimal solution. Indeed, a kind of 'bang-bang' phenomenon occurs, in that the chosen policies possess sharp discontinuities at  $x_{10}$ , as shown in Fig. 2. The example was then run with the *convex* building-cost function  $c(u) = u^{1.20}$ . [Examples of problems with fixed delays and convex objective function can be found in education, since the cost of education is typically a

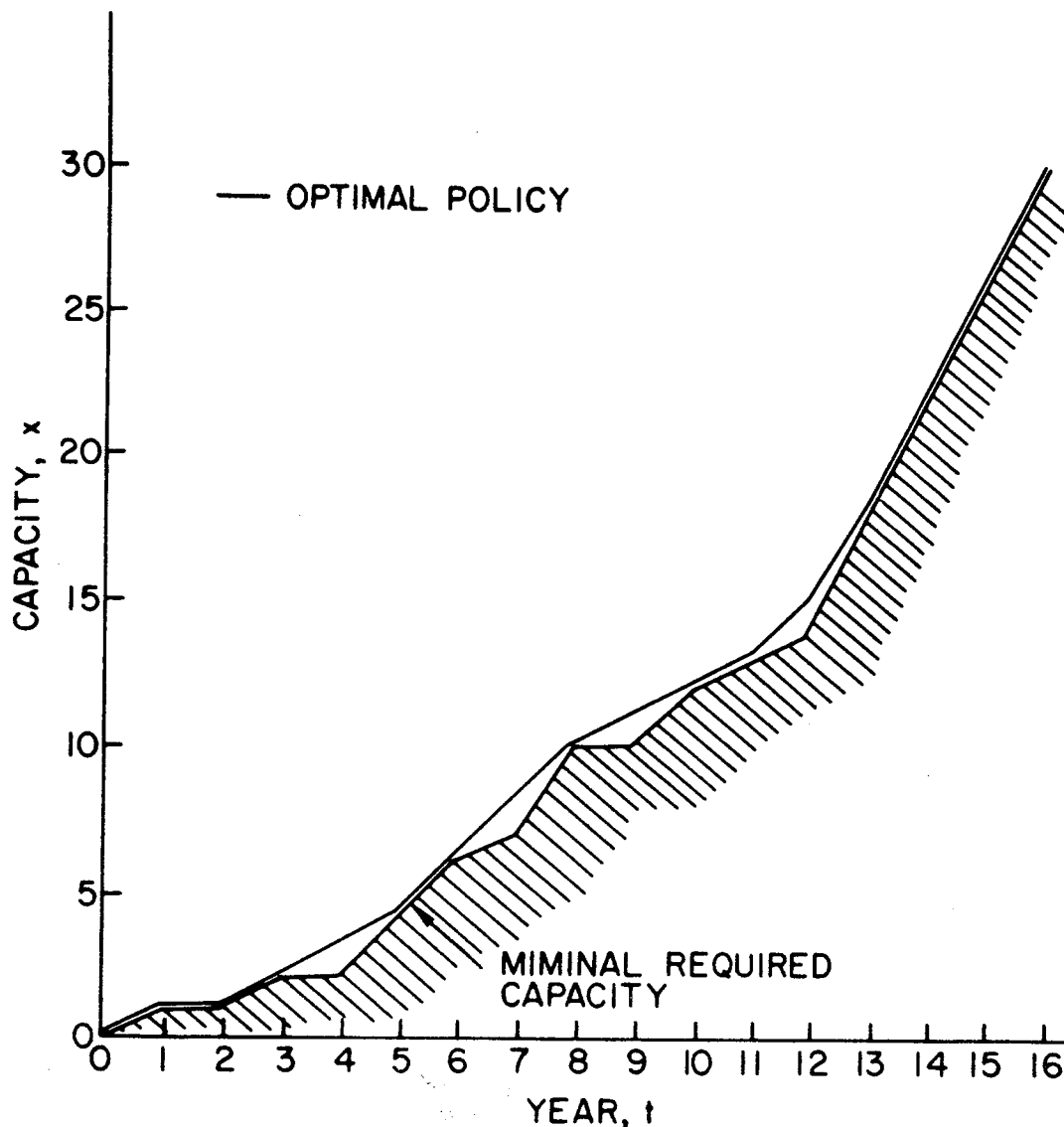


Figure 3

convex function of the number of students involved, and since the replacement of faculty members introduces a fixed delay of about 20 years.]

Here the Lagrange-multiplier procedure is applicable and the optimal solution was determined after the first real iteration (see Fig. 3). Indeed, it can be shown that the optimal solution is obtained with any  $\lambda$  in the range  $0.455 \leq \lambda \leq 0.5005$ .

### CONCLUSION

WITH THE cyclic-dynamic-programming method, exact solutions to problems with fixed delays can be obtained within reasonable computer times. In general, some kind of iteration of a basic dynamic program over the possible boundary conditions is required. Under appropriate conditions (convexity of the objective function for example) this iteration can be performed over a Lagrange-multiplier vector.

### ACKNOWLEDGMENTS

THE AUTHOR is indebted to ANTOINE DELARUE for valuable discussions and assistance during the course of this research, which was supported by the National Science Foundation.

### REFERENCES

1. R. BELLMAN AND S. DREYFUS, *Applied Dynamic Programming*, Princeton Univ. Press, Princeton, N. J., 1962.
2. R. BROOKS AND A. GEOFFRION, "Finding Everett's Lagrange Multipliers by Linear Programming," *Opns. Res.* 14, 1149-1153 (1966).
3. H. EVERETT, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Opns. Res.* 11, 399-417 (1963).
4. L. LASDON, "Duality and Decomposition in Mathematical Programming," *IEEE Trans. on Systems Science and Cybernetics*, SSC-4 No. 2, July 1968.
5. D. LUENBERGER, *Optimization by Vector Space Methods*, Chaps. 8, 9, and 10, Wiley, New York, 1969.