M. K. Goel

TRAINING COURSE

ON

RESERVOIR OPERATION

(UNDER WORLD BANK AIDED HYDROLOGY PROJECT)

Module 10

DP and Its Applications to WRS

BY

S K Jain, NIH
M K Goel, NIH

NATIONAL INSTITUTE OF HYDROLOGY
ROORKEE - 247 667, INDIA

DP AND ITS APPLICATIONS TO WRS

1.0 INTRODUCTION

Dynamic programming (DP) is an enumerative technique developed by Richard Bellman in 1953. This technique is used to get the optimum solution to a problem which can be represented as a multistage decision process. The entire formulation of dynamic programming is based upon the Bellman's principle of optimality. According to this principle, an optimal policy has the property that whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision. In the Fig. 1, if the optimal path for going from A to D is ABCD then the optimal path from B to D will be BCD and not BED.

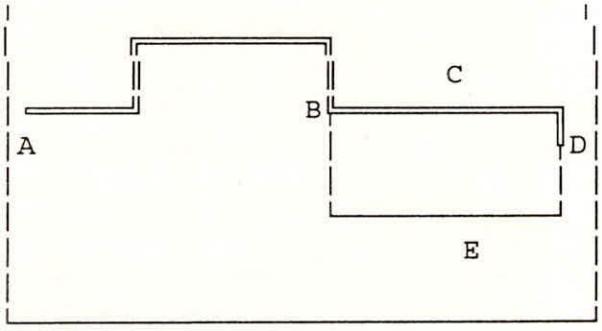


Fig. 1 Illustration of the Principle of Optimality

Dynamic programming is not a class of optimization technique, but as an algorithm it is a very powerful procedure developed to solve sequential decision problems. Many problems in water resources, such as reservoir operation, involve a sequence of decisions from period to period and can be solved by dynamic programming.

In a Dynamic Programming problem formulation, the dynamic behavior of the system is expressed by using three types of variables, as described below :

State Variables - which define the condition of the system. For example, in studies dealing with reservoirs, the amount of water stored in the reservoir may represent its state.

Stage Variables - which define the order in which events occur in the system. Most commonly, time is taken to be the stage variable. There must be a finite number of possible states at each stage.

Control Variables - which represent the controls applied at a particular stage and transform the state of the system. For the reservoir operation problem, the release of water from the reservoir is a typical decision variable.

The dynamic behavior of the system is expressed by an equation known as the system equation. It can be written in discrete form as :

$$s(t+1) = f[(s(t), u(t),t] \qquad t = 1, 2.....N \qquad ...(1)$$

where s(t) is the state variable at time t, u(t) is the control applied at time instant t, which last for a duration t and f() is the given functional form.

The state of the system at any stage should lie in the domain of admissible states at that stage. The controls at stage t should also lie in the admissible domain at that stage:

$$s(t) \epsilon S(t), \qquad u(t) \epsilon U(t)$$

where S(t) and U(t) are the domains of admissible states and controls at stage t.

With each state transformation, a return is associated which may either represent benefits or costs. Typically the benefits are maximized and the costs are minimized. The optimal decision made at a particular stage is independent of decisions made at previous stage given the current state of the system. A set of decisions for each time period is called a policy. The policy which optimize the objective function is called the optimal policy. The set of states which result from the application of a policy is called the state trajectory. As an example, the volume of water stored in a reservoir can be considered to be its state. The state of a reservoir is transformed due to inflows and can be controlled by releasing water from the storage. This water can be used for some useful purpose (irrigation for example) to yield monetary returns or it may also cause flood damages downstream and a cost is associated with these damages. The problem is to find the releases (controls) which optimize the returns.

2.0 THE SOLUTION ALGORITHM

Let R[s(t), u(t), t] be the return obtained if the system is at state s(t) at stage t and the control u(t) is applied at instant t lasting for a duration t. Further, let F[s(N),N] be the sum of returns from application of controls from some initial stage at t = 0 to final stage at t = N. The objective of maximizing the sum of returns from the system can be expressed as

$$\text{Max } F[s(N), N] \qquad ...(2)$$

Let the state of system at t = 0, s(0) ϵ S(0) is known and the returns F[s(0),0] are also known. Let F[s(0),0] be the optimum value of these returns. Now consider the first stage (of duration t). The optimal return for this period is given by

$$F[s(1),1] = \underset{u(t-1) \ \epsilon \ U(t-1)}{\text{Max}} \ R[s(0),u(0),0] + F[s(0),0] \qquad ...(3)$$

This equation is solved for each discrete level of state at t=1 as a function of control variables u(0). To do this, the state is discretized into a number of discrete levels. Now a particular lattice point

is chosen and all the admissible levels of decision variables which lead to this state are chosen. For each of these decision variables, the return F[s(1),1] is calculated. The maximum among these returns given the value of F[s(1),1]. This computation is repeated for each discrete value of s(1) and the results are stored.

The computations are performed in similar fashion for stage 2,3....N. The recursive equation for any stage t can be written as:

$$F[s(t),t] = \underset{u(t-1) \in U(t-1)}{\text{Max}} R[s(t-1),u(t-1),t-1] + F[s(t-1),t-1] \quad \ldots(4)$$

Thus at the end of N stage, the values of F[s(t),t], t=1,2...N are available. The optimal value of control variables or the optimal policy is obtained by tracing back the values of returns, corresponding to those states which satisfy the initial and final values and the constraints. The optimal state trajectory can be determined by using the system equation once the optimal policy is known.

The above computational scheme of dynamic programming is known as the forward algorithm since the computations start at the initial value of the state variable at stage 1 and move forward. In contrast to this, the computations can also commence at the final value of state variable at the last stage and can move backwards. The optimal policy is retrieved by tracing forward from the returns. This algorithm is called the backward algorithm.

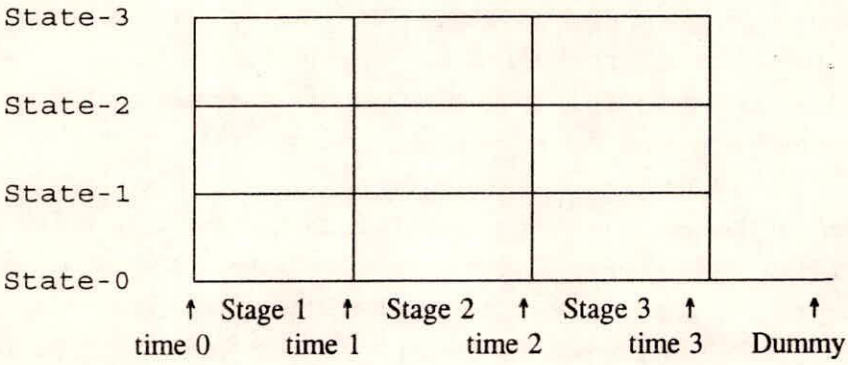
The constraints which restrict only the state or decision space are advantageous in DP because they reduce the amount of computation. In contrast, state and decision space constraints can cause considerable procedural difficulties for other optimization techniques. However, when DP is applied to a multiple reservoir system, the usefulness of the technique is limited by the so-called curse of dimensionality which is a strong function of the number of the state variables. For computational efficiency, the problems should have few state variables at a time. All the methods of dimensionality reduction involve decomposition of the system into subsystems and use of iterative procedures.

Example: The algorithm is illustrated through an example. A person has 3 Rupees which he plans to spend over a period of 3 days such that all the money should be spent at the end of day 3. We limit our analysis to integer increments of rupee only. The benefit that the person gets by spending various amounts of money on each day is given in the following table :

Return function R(u) values for various stages and decisions

		Stage 1	2	3
Decision	0	0	0	0
	1	3	1	4
	2	7	2	6
	3	10	4	8

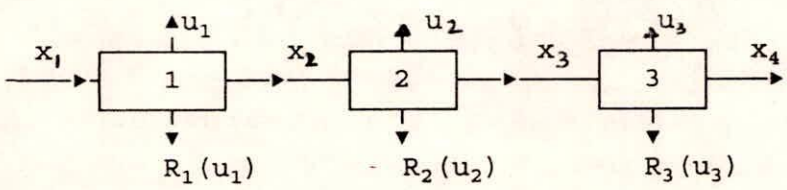
Note that in this example, the returns are a function of decision (u) only. In general, they can be a function of both decision (u) and the state of the system (x). Also note that the returns for spending money are not linear or continuous functions. It is very convenient to visualize the problem in a rectangular network consisting of nodes at the grid points and arcs connecting them. Each node can be assigned the state attained by the system and all the feasible states attained by the system at a particular time can be arranged on one vertical line. Each such vertical line represents a stage. Any movement from any node at one stage to any other node at the next stage along the corresponding arc represents a state transition and thereby dictate the return resulting from such a decision. Consider the network shown below. The return function is also shown for various discrete decisions 0, 1, 2, 3.



For understanding purposes, let x represent the money on hand, u the money spent on any day, each stage represents a day and the return function is the enjoyment expressed in numerical units derived by spending money on any day. The objective is

$$\text{Max } Z: \; R_1(u_1) + R_2(u_2) + R_3(u_3) \qquad \ldots(5)$$

Let $f_i(x_i)$ represent the objective function value at the i^{th} stage where the system state is x_i. The sequential process is illustrated in the figure below:



Let us start the analysis for day 3. Now, $f_4(x_4) = 0$, i.e. the total return expected in the remaining time after day 3 being at state zero is zero

$$f_3(x_3) = \text{Max } [R_3(u_3) + f_4(x_4)]$$

This computation for stage 3 can be organized in the following table.

x_3	u_3	x_4	$R_3(u_3)$	$f_4(x_4)$	$f_3(x_3)$	
0	0	0	0	0	0	
1	0	1	0	0	0	infeasible
	1	0	4	0	4	optimal policy.
2	2	0	6	0	6	
3	3	0	8	0	8	

The value of u_3 represents each possible action on day 3 and as illustrated for $x_3 = 1$, the best possible action at stage 3 is to set $u_3 = x_3$.

For Day-2: The return for days 2 and 3 is $f_2(x_2) = \text{Max } [R_2(u_2) + f_3(x_3)]$ and $x_3 = x_2 - u_2$. The computations for this stage are given below

x_2	u_2	x_3	$R_2(u_2)$	$f_3(x_3)$	$f_2(x_2)$
0	0	0	0	0	0
1	0	1	0	4	4*
	1	0	1	0	1
2	0	2	0	6	6*
	1	1	1	4	5
	2	0	2	0	2
3	0	3	0	8	8*
	1	2	1	6	7
	2	1	2	4	6
	3	0	4	0	4

* indicates the optimum policy

For Day-1: The return for day 1, 2 and 3 is $f_1(x_1) = \text{Max } [R_1(u_1) + f_2(x_2)]$ and $x_2 = x_1 - u_1$. The computations for stage 1 are given below

x_1	u_1	x_2	$R_1(u_1)$	$f_2(x_2)$	$f_1(x_1)$
0	0	0	0	0	0
1	0	1	0	4	4*
	1	0	3	0	3
2	0	2	0	6	6
	1	1	3	4	7*
	2	0	7	0	7*
3	0	3	0	8	8
	1	2	3	6	9
	2	1	7	4	11*
	3	0	10	0	10

* indicates the optimum policy.

Now we can trace back the optimum policy for any level of x at the beginning of day 1. For example if x = 3 at time = 0, then max return = 11.

From table for day 1 : $u_1 = 2$, $x_2 = 1$
From table for day 2 : $u_2 = 0$, $x_3 = 1$
From table for day 3 : $u_3 = 1$.

Observe that this computation was carried out in a reverse direction to that of time and so it is referred to as backward computation. We could also solve the problem starting from day 1. The backward computation is used when there are known end conditions and we need solutions for every starting condition. This was the situation in the example and that is why we used backward dynamic programming. The forward computation is used when there is a starting condition and we need solutions for every ending condition. Where there is no special reason for choosing either backward or forward formulation, the backward recurrence is normally used. The procedure of making first a backward and then a forward pass is convenient especially in problem involving time as it gives the optimal policy in the chronological order.

3.0 APPLICATION OF DP TO RESERVOIR OPERATION

Let us consider the problem of determination of optimum release from a reservoir. If DP is applied for the determination of reservoir release, the state variable is the storage and the decision variable is the release. The stage is represented by the time period i. The stage-to-stage transformation is characterized by the continuity equation

$$S_{i+1} = S_i + I_i - R_i - e_i \qquad \ldots(6)$$

subject to:

$$S_{min} \leq S_i \leq S_{max}$$

Suppose an objective function J(S, R) has been chosen for maximization. Note that J, in general, is a function of release as well as storage. A typical forward DP recursive equation can be written as:

$$f_{i+1}(S_{i+1}) = \max [J (R_i, S_i) + f_i(S_i)], \qquad i = 0, 1, 2, \ldots\ldots, T \qquad \ldots(7)$$

given initial storage S_0. The state variable (storage) is discretized into a number of feasible states. Suppose that the inflow sequence is given and the evaporation term is temporarily ignored, the continuity equation becomes

$$S_{i+1} = S_i + I_i - R_i \qquad \ldots(8)$$

If S_{i+1} and S_i are chosen, R_i can be directly computed from the above continuity equation. The optimization is over the proper choices of R 's. The problem of interpolation is avoided, since

the R values are computed by fixing the states S_i and S_{i+1}. Solutions are imbedded in the discretized states. The infeasible transitions are discarded in the solution process. The inclusion of the evaporation term poses no difficulty, since evaporation is a function of the average storage S which is equal to (S_i + S_{i+1})/2. The recursive equation is carried out until the final stage T is reached. The optimal solutions can then be traced back to determine the consequent release and storage.

In summary, the use of dynamic programming for analysis of reservoir design and operation, has three important advantages compared to linear programming; first and most important is the ability of dynamic programming to easily handle non linear objective functions and constraints. The second advantage is that it is relatively easy to solve the dynamic programming problem for many months or periods. The third advantage is that constraints decrease the computational burden of dynamic programming.

4.0 SCREENING OF RESERVOIRS IN A RIVER BASIN USING DP

In the planning of storage projects in a river basin, one of the major problems is to find out which reservoirs and of what sizes are to be selected from an inventory of reservoir sites. This information may be required for a given level of development, in terms of any indicator like the amount of yield to be obtained or area to be irrigated from the set of reservoirs considered.

It is evident that several combinations of reservoirs of various size may exist in a river basin that may provide the same total yield of the system. A crude method would be to estimate the costs of all these possible alternatives and to retain the alternative with the least cost. As the number of possible alternatives can be extremely large, such a procedure would necessitate a great amount of computation which may be expensive. The dynamic programming is a method by which the number of investigated alternatives can be considerably reduced, without omitting any relevant alternative.

Though the yield of a reservoir or of the entire system is a continuous variable from 0 to any value, the value of the yield is discretised. This implies selection of an increment of yield for discretising the function. For example, if the increment is one cubic meter, then discretised value of yield can be 0, 1, 2 , 3 ....n cubic meter. If 10 cubic meter is the increment, then discretised values becomes 0, 10, 20, ... n cubic meter etc. Only these discretised values are used in the optimization process. The discretised values of the system yield are referred to as the states of the system.

For the optimization process, various reservoirs are presented in a sequence and an order number is assigned to each reservoir. The dynamic programming sequentially add one reservoir at a time into the system, in the order of numbers assigned to the various reservoirs of the system. Thus, successively optimal combinations of each additional reservoir with the previously combined system are investigated. A combination of each additional reservoir with the previously combined system is referred to as a stage of the analysis. This is illustrated by an example in this section.

The decision to be taken at each stage is how much yield is to be derived from the reservoir

to be added now. Thus the decision variable is the yield from the presently considered reservoir. The objective of the problem can be to minimize the cost of providing the required yield. Constraints of the formulation are also to be incorporated in the analysis. One can be that for each stage, the yield from all individual reservoirs should add up to the desired yield. Another can be that the yield from any reservoir can not exceed the maximum possible yield. State transition function is derived from the fact that the yield from previous stage plus the yield from the additional reservoir should add up to the yield required from the system. Illustration of the Dynamic Programming procedure, as given below, will enable better understanding of the problem:

Example: A system of three reservoirs is considered with following data. It is required to find the optimum yield combination from each reservoir for getting a total system yield of 60.

Reservoir 1		Reservoir 2		Reservoir 3	
Yield	Cost	Yield	Cost	Yield	Cost
0	0	0	0	0	0
20	15	20	10	20	20
40	30	40	35	40	40

At stage 1, only reservoir 1 is considered. The costs of providing various yields (in the range 0 to 40) will be equal to the cost of yields provided by reservoir 1. At stage 2, any combination of reservoirs 1 and 2 giving the required yield would be feasible. The optimum yield configuration table from reservoir 1 and reservoir 2 for all range of yield from 0 to 80 is given in the following:

Total	Yield from		Cost of		Total
Yield	Reser. 1	Reser. 2	Reser. 1	Reser. 2	Cost
0	0	0	0	0	0
20	20	0	10	0	10
40	20	20	15	10	25
60	40	20	30	10	40
80	40	40	30	35	65

Now, for a total system yield of 60, the possible combinations and corresponding costs are given in following table:

Yield from		Total	Cost of		Total
Reser. 3	Stage 2	Yield	Reser. 3	Stage 2	Cost
0	60	60	0	40	40
20	40	60	20	25	45
40	20	60	40	10	50

Hence for a total system yield of 60, yield of 40 from reservoir 1 and yield of 20 must be obtained from reservoir 2. There is no need to construct reservoir 3 in this case. Similarly, for a total system yield of 80, yield of 40 must be planned from reservoir 1, yield of 20 from reservoir 2 and yield of 20 must be planned from reservoir 3.

5.0 ADVANTAGES AND DISADVANTAGES OF DYNAMIC PROGRAMMING

Dynamic programming is essentially an enumerative technique which is specially suited to multistage decision problems. There are a number of advantages in using this technique for analysis of a water resources system. Some of the advantages are :

i) The dynamic programming formulation is same for linear as well as nonlinear problems. Thus, no extra effort is required for nonlinear problems. This property is very useful in case of water resources systems since many related problems can not be realistically linearized.

ii) The incorporation of constraints in linear and nonlinear programming problems is more difficult than in dynamic programming problems. In case of dynamic programming, the constraints serve a useful purpose. They do limit the feasible region and thus many lead to reduction in computational time requirement.

iii) The stochastic nature of a problem can be easily considered in the dynamic programming formulation. The algorithm developed for a deterministic problem does not have to be significantly changed to incorporate stochasticity. This is in contrast with other techniques where incorporation of stochasticity requires too much change in the algorithm and significant increase in computational time.

Along with the above advantages, there are some disadvantages also :

i) The dynamic programming is not basically tailored in such a fashion that generalized programs can be written using it. Thus a new computer program has to be developed or an existing program has to be significantly modified and tested for each new application of the technique. On the other hand standard computer programs are widely available for the linear programming.

ii) It was stated above that to solve a particular problem, the state and control variables are discretized at each stage and these discretized values are then used. This approach is known as the Discrete Dynamic Programming(DDP) technique in which the state and control spaces are discretized by finite sets of vectors. For each stage and state, the continuous variables are replaced by the discrete node points and these values have to be stored in the computer memory from where they can be drawn as and when required. The number of discretized values goes on increasing with the fineness of the discretization. For large problems, the memory requirement becomes a major limitation. This requires judicious choice to be made for the accuracy requirement, computer memory available and computational time available.

Several techniques have been proposed by different investigators to reduce the dimensionality problem associated with analysis of water resources systems using dynamic programming technique. These include the Incremental DP (IDP) and the Discrete Differential Dynamic Programming (DDDP). The basic approach in these techniques is the same and their are only minor differences regarding the increments in state and stage variables. It may be noted that these schemes are some sort of *successive approximation* schemes. An initial estimate of the policy is made and this is used to construct an improved estimate. This improved estimate becomes the input to the next stage and so on until some convergence criterion is satisfied. The scheme can not assure the global optimum and may converge to a local optima. However, by starting from different initial solutions, the possibility of finding the global optimum is increased.

The technique of dynamic programming has been described in a number of excellent texts, some of these are given in references. Yakowitz(1982) has reviewed the applications of DP to water resources. Yeh(1985) provides an excellent review of applications of DP to reservoir operation.

6.0 REFERENCES

Hall,W.A., and J.A. Dracup, Water Resources Systems Engineering, Tata McGraw-Hill Publishing Company, New Delhi, 1979.

Loucks, D.P., J.R. Stedinger, and D.A. Haith, Water Resources Systems Planning and Analysis, Prentice Hall Inc., New Jersey, 1981.

Rao, S.S., Optimization, Theory and Practice, Wiley Eastern, 1979.

Yakowitz,S., "Dynamic programming applications in water resources", Water Resources Research, 18(4), 673-696, 1982.

Yeh, William W-G., "Reservoir management & operation models : A state of the art review", Water Resources Research, 21(12), 1797-1818, 1985.

***