A Network Flow-Dynamic Programming Algorithm for Complex Water Resevoir Problems

J. Casti

October 1974 WP-74-52

A Network Flow-Dynamic Programming Algorithm for Complex Water Resevoir Problems

J. Casti

1. Introduction

A problem of basic importance in the computational analysis of complex water reservoir networks is to account for stochastic inflows and nonlinear objectives without dramatically altering the computational complexity (and cost) of the problem. Without the stochastic/nonlinear features, network flow algorithms provide extremely efficient and effective procedures for the problem solution. On the other side of the coin, dynamic programming procedures enable one to include virtually any realistic feature but at the added expense of greatly increased computing time and storage requirements. As usual, the water resource analyst is perched on the horns of a dilemma--cheap solutions with the dangers inherent in oversimplification versus accurate solutions at great expense and effort.

In this note, we propose an algorithm which represents a compromise between the two extremes. Network flow analysis is employed to effect local optimizations and then dynamic programming ideas are introduced in order to piece the local solutions together into an optimal global policy. In this way, it is hoped to make maximum use of the best features of each method: the speed of the network flow techniques and the broad generality of dynamic programming.

2. The Basic Problem

Let us consider a system in which there are k reservoirs which must be controlled in an optimal fashion over N time periods. Let

$s_i(t)$ = the level of water in reservoir i at time period t,

$u_i(t)$ = the amount of water released from reservoir i at time period t,

$i = 1,2,\ldots,k \quad , \qquad t = a,a+1,\ldots,N \quad .$

On the basis of water use for recreation, power generation, flood control, irrigation, and navigation, constraints both on the level of water in the reservoir at any given time and the amount of water released during any period are imposed upon the variables $s_i(t)$ and $u_i(t)$. These constraints are of the form

$$\begin{aligned} &\sigma_i \le s_i(t) \le S_i \quad , \\ &\mu_i \le u_i(t) \le U_i \quad , \qquad i = 1,2,\ldots,k \quad , \end{aligned} \tag{1}$$

where σ_i, S_i, μ_i,U_i, are given constants.

As a consequence of the release of an amount of water $U_i(t)$ and an initial level of water $s_i(t)$, a certain benefit $\ell_i(s_i,u_i)$ is obtained and a new water level $s_i(t + 1)$ is achieved according to the system dynamics

$$s_i(t + 1) = s_i(t) - u_i(t) + r_i(t) \quad , \tag{2}$$

$$s_i(a) = c_i \quad ,$$

where $r_i(t)$ is a random variable denoting the stochastic inflow to reservoir i at time t. This quantity is due to rainfall and various seepage effects into the river basin. For simplicity, assume that the variables $r_i(t)$ are independent and identically distributed with distribution function $dG(r,t)$.

If we agree that system performance is to be measured in terms of the total expected gain, then the problem is to maximize the expected value of

$$J = \sum_{t=a}^{N} \sum_{i=1}^{k} \ell_i \; s_i(t), u_i(t) \quad ,$$

subject to the dynamics (2) and the constraints(1).

3. Dynamic Programming Formulation

The problem sketched in the preceding section could be easily resolved by standard network flow techniques <u>if</u> the functions $\ell_i(s,u)$ were linear in s and u and <u>if</u> the stochastic forcing terms in (2) were absent. Unfortunately, in real systems both the nonlinearity of the ℓ_i's and the stochastic inputs are integral parts of the process and cannot be ignored. Consequently, the linear programming-type algorithms are stretched beyond their breaking point and recourse must be taken to more general techniques. The usual approach is to employ dynamic programming to take advantage of this method's suitability for handling a wide range of realistic features.

Briefly, the procedure is as follows: let

$f_a(c_1, c_2, \ldots, c_k)$ = expected value of J when N-a periods remain, the reservoirs have levels c_i, and an optimal water release policy is used, $i = 1,2,\ldots,k$, $a = N, N-1, \ldots, 0$.

As a result of <u>any</u> initial decision $u_i(a)$, the water levels c_i change according to (2) and an expected return

$$\int \sum_{i=1}^{k} \ell_i\Big(s_i(a), u_i(a)\Big)\, dG(r,a)$$

is obtained. According to the Principle of Optimality, the decision $u_i(a)$ must be made in an optimal fashion, if any policy including $u_i(a)$ is to be optimal. Putting all these remarks together, we obtain the functional equation

$$f_a(c_1, c_2, \ldots, c_k) = \max_{\mu_i \leq u_i(a) \leq U_i} \left\{ \int \left[\sum_{\tau=1} \ell_i(c_1, \ldots, c_k, u_1, \ldots, u_k) \quad f_{a+1}(c_1 - u_1 + r_1, \ldots, c_k - u_k + r_k) \right] dG(r,a) \right\}$$

$$\sigma_i \leq c_i \leq S_i \quad ,$$

$$a = N-1,\ N-2, \ldots, 0 \quad ,$$

$$i = 1, 2, \ldots, k \quad ,$$

with the initial condition

$$f_N(c_1,c_2,\ldots,c_k) = \phi(c_1,c_2,\ldots,c_k) \quad ,$$

where ϕ is a function measuring the benefit of having water levels $c_1,c_2,\ldots c_k$ which cannot be released.

4. Numerical Solution

Solution of the foregoing functional equation may be easily resolved if k is small, e.g. $k = 2$ or 3. However, for large systems in which k may be 8 or 10, serious computational difficulties arise due to the so-called "curse of dimensionality." If, for example, each c_i may assume 10 values and $k = 6$, then the function $f_a(c_1,\ldots,c_k)$ must be evaluated <u>and stored</u> for 10^6 different values of its argument for each a, and for each evaluation a minimization over the u_i's must be carried out. Consequently, any straightforward "search and store" procedure for computing f must be ruled out if k is of realistic size. In order to make progress, additional ideas and techniques will have to be employed.

To cut the computational requirements down to size, we shall simultaneously employ two approximation techniques. First of all, we utilize approximation in policy space. For ease of notation, we write the functional equation for f as

$$f_a(c) = \max_{u\in\mathscr{U}}\left[\int L(c,u) + f_{a+1}\Big(T(c,u,r)\Big)\, dG(r)\right] \quad ,$$

where c and u are vector quantities, and $\mathscr{U}$, L, T have obvious meanings. Then approximation in policy space proceeds as follows:

i) guess an initial admissible policy $u_a^0(c)$

ii) calculate $f^{(0)}$ by

$$f_a^{(0)}(c) = \int \left[L(c,u^0) + f_{a+1}^{(0)}\left(T(c,u^0,r)\right)\right] dG(r) \quad ,$$

$$f_N^{(0)}(c) = \phi(c) \quad , \qquad a = N-1,\ N-2,\ldots,0 \quad ;$$

iii) determine the next approximation $u_a(c)$ as that policy which maximizes the quantity

$$\int \left[L(c,u) + f_{a+1}^{(0)}\left(T(c,u,r)\right)\right] dG(r) \quad , \qquad a = N-1,\ldots,0 \quad ;$$

and repeat steps ii) and iii) until convergence takes place.

Under very reasonable hypotheses on L, it can be shown that the above procedure is always convergent, in fact, monotonically convergent ($f_a^{(0)}(c) \le f_a^{(1)}(c) \le \ldots$ pointwise in c for each a).

The difficulty with implementing the policy space interation procedure for high-dimensional processes is in carrying out the maximization in step iii). However, in the water resource problem the function L(c,u) has special structure which may be employed to reduce the maximization process to one which may be carried out by the efficient network flow algorithm. The basic structure we exploit is separability and concavity of the functions $\ell_i(c,u)$, and the fact that the control u does not enter, i.e.

$$\ell_i(c,u) = \ell_i(c_i) \quad , \qquad i = 1,2,\dots,k \quad ,$$

$$\frac{d^2}{dc^2} \ell_i(c_i) < 0 \quad .$$

Thus, we may approximate each function $\ell_i(c,u)$ as

$$\ell_i(c_i) = \begin{cases} a_{1i}c_i + b_{1i} \quad , & 0 \leq c_i \leq C_1 \quad , \\ a_{2i}c_i + b_{2i} \quad , & C_1 < c_i \leq C_2 \quad , \\ a_{ni}c_i + b_{ni} \quad , & C_n < c_i \leq C_{n+1} \quad , \\ & i = 1,2,\dots,k \quad . \end{cases}$$

At the same time, we may also use a multiple linear regression technique to approximate the return function

$$f_{a+1}^{(m)}(c_1 - u_1 + r_1,\dots,c_k - u_k + r_k)$$
$$\simeq \sum_{i=1}^{k} \alpha_i(c_i - u_i + r_i) + \beta \quad .$$

Having made the foregoing approximations, the minimization in step iii) becomes a network flow problem which may be solved by the usual techniques. Notice that the approximation of the ℓ_i need be done only once, while the multiple regression approximation is carried out for each a.

Let us now summarize the proposed algorithm:

0. Approximate the functions $\ell_i(c_i)$ by piecewise linear functions calling the approximation $\tilde{L}(c,u)$ and guess an initial policy $u_a^{(0)}(c)$.

1. Compute $f_a^{(0)}(c)$ by iterati

$$f_a^{(0)}(c) = \int \left[L(c,u^0) + f_{a+1}^{(0)}\left(T(c,u^0,r)\right)\right] dG(r) \quad ,$$

$$f_a^{(0)}(c) = \phi(c) \quad .$$

2. Approximate $f_a^{(0)}(c_1 - u_1 + r_1,\ldots c_k - u_k + r_k)$ by linear functions of the arguments. Call the approximation $\tilde{f}_a^{(0)}\left(T(c,u,r)\right)$.

3. Determine $u_a^{(1)}(c)$ by employing network flow analysis to maximize

$$\int \left[\tilde{L}(c,u) + f_{a+1}^{(0)}\left(T(c,u,r)\right)\right] dG(r) \quad .$$

4. Repeat steps 1 - 3.