Data Management for Large-Scale Water-Distribution Optimization Systems

By S. Pezeshk,[1] Associate Member, ASCE

Abstract: This paper presents a technique to efficiently process data to supply the necessary in-puts to a large-scale water-distribution optimization model. The overall goal of this project is to develop an on-line optimal control strategy for a large municipal and industrial (M&I) water distribution system. That is, which pumps should be turned on to meet the demand and minimize cost? One of the problems in handling a large water-distribution system is managing the large database required for the optimization model. This paper presents an efficient algorithm to generate, store, and access the necessary data. Some of the characteristics of the method are illustrated through an example of a water-distribution system.

INTRODUCTION

This paper presents a technique to efficiently process data for input to a large-scale water-distribution optimization model. The overall goal of this project is to develop an on-line optimal control strategy for a large municipal and industrial (M&I) water-distribution system. That is, which pumps should be turned on to meet the demand and minimize cost?

A number of investigators have dealt with the problem of optimal water-distribution systems. Some have utilized both simulation models and optimization models, while others have used one or the other. Our approach is to rely on a large-scale optimization model called MINOS (Murtagh and Saunders 1983) to generate optimal control, and use the simulation model, KYPIPE (Wood 1988), to check feasibility and generate constraints for the optimization model.

One of the problems in this approach is managing the large database required for the optimization model. What is the best way to generate, store, and access the necessary data? This paper represents an efficient algorithm to do that.

FORMULATING DATA PREPROCESSOR

The general form of the nonlinear optimization model is

$$\text{Minimize} \quad \sum_{i=1}^{n\_pump} \frac{TDH_i}{\eta_i} Q_i \quad \text{(1)}$$

$$\text{Subject to} \quad \sum h_L = \sum E_p \quad \text{(2)}$$

$$\sum Q_{\text{in}} - \sum Q_{\text{out}} = Q_e \quad \text{(3)}$$

where TDH (total dynamic head) $= AQ^2 + BQ + C$ in which A, B, and C are constants for a given pump; n_pump = total number of pumps in the system; η = pump efficiency; Q_{in} = flow into the junction; Q_{out} = flow

[1]Asst. Prof., Dept. of Civ. Engrg., Memphis State Univ., Memphis, TN 38152.

Note. Discussion open until July 1, 1994. To extend the closing date one month, a written request must be filed with the ASCE Manager of Journals. The manuscript for this technical note was submitted for review and possible publication on November 30, 1992. This technical note is part of the ***Journal of Water Resources Planning and Management***, Vol. 120, No. 1, January/February, 1994. ©ASCE, ISSN 0733-9496/94/0001-0116/$1.00 + $.15 per page. Technical note No. 3553.

J. Water Resour. Plann. Manage. 1994.120:116-120.

out from the junction; Q_e = external inflow or demand at the junction node; h_L = energy loss in each pipe; and E_p = energy put into the system by a pump. Eq. (2) is the loop constraint that requires the sum of head losses in any loop to equal zero, analogous to Kirkouff's Law. Eq. (3) is the conservation of mass, which requires the sum of flows entering a node to equal the flows exiting the node. To satisfy the data requirements of these two constraints, a considerable amount of "book keeping" is required. For example, the data set needs to specify which pipes are connected to which nodes and which pipes are in which loops. To satisfy these requirements, we developed a system of pointers and vectors to process the data and to in-put them to the optimization model.

To illustrate the procedure, consider the simple water-distribution system shown in Fig. 1, which consists of eight nodes, eight pipes, three pumps, and one reservoir. In processing a network such as the one given in Fig. 1, the required information may be found in the basic connectivity matrix **C**, which relates the nodes and pipes [see (4)].

$$\mathbf{C} = \begin{array}{c} \text{Node Numbers} \\ \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} +1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ -1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & +1 \\ 0 & 0 & +1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \quad \text{Pipe Number} \qquad \ldots\ldots\ldots (4)$$

The positive numbers represent the direction of flow away from the node, and the negative numbers represent the direction of the flow into the node. Since each pipe has only two nodes at each end, there are only two nonzero elements in each row of matrix **C**.

Each row of matrix **C** represents a pipe number and each column represents a node number. The size of the **C** matrix is *n_pipe* × *n_node*, where

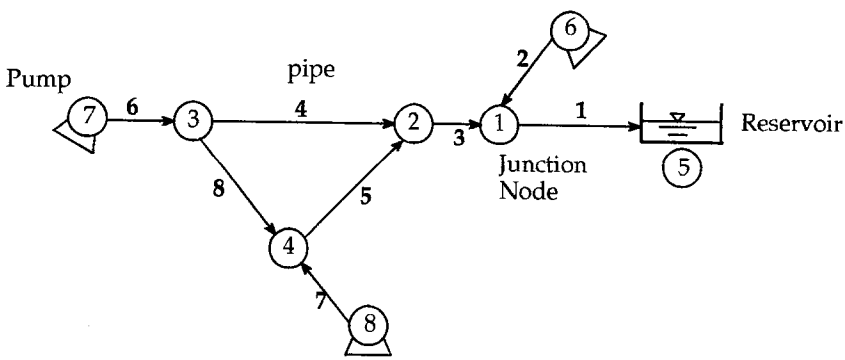

FIG. 1. Topology of Water-Distribution Network

J. Water Resour. Plann. Manage. 1994.120:116-120.

n_pipe is the total number of pipes and *n_node* is the total number of nodes in the network. For a large water-distribution network there could be as many as 2,000 pipes and 2,000 nodes. In this case, the size of the **C** matrix will be 2,000 × 2,000, with only 4,000 nonzero elements and 3,996,000 zero elements. It is obvious that a large amount of space is occupied by extraneous data, which will exceed the storage and memory capacity of many computers. Furthermore, to answer the questions posed earlier, one needs to perform a search among 2,000 × 2,000 components of a matrix for each individual case, making it computationally prohibitive.

A much more efficient way of storing the **C** matrix would be to store only the nonzero elements. The **C** matrix can be easily stored as **c_comp**, which has the dimension of *n_pipe* × 2 (see Fig. 2). The first column of the matrix represents the node number at the beginning of a pipe, and the second column represents the node number for the second node of the pipe. For example, one can see that pipe 3 runs from node 2 to node 1. The direction of the flow is from the first node to the second node. The matrix **c_comp** can efficiently and effectively be used to determine which nodes are connected to a given pipe.

There are two data items required: given a pipe, what nodes bracket the pipe, and given a node, what pipes are connected to it? The first item is easily answered from the **c_comp** matrix. Enter the **c_comp** matrix with the pipe number and read the corresponding nodes. The order of the nodes indicate the direction of flow in the pipe.

The second item requires two additional vectors, **ipointer**, with dimensions of (*n_node* + 1) × 1, and **ipipe** with dimensions of (2*n_pipe* + 1) × 1. These two vectors can be easily constructed from the matrix **c_comp** . To generate the final **ipointer**, one needs to generate a temporary vector **itemp** from **c_comp** matrix. The *i*th component of the **itemp** vector would be the number of the branches incident on node *i*, and can be generated with one pass over the **c_comp** matrix. When **itemp** is available, the **ipointer** vector can be constructed from **itemp**. For example, the first component of **itemp** will be three because there are three incidences of one in the **c_comp** matrix, which means that there are three branches connected to node one. Consequently, the **ipointer** vector will have a value of 3 + 1 = 4 in the second component. The **ipipe** vector can be generated directly from **ipointer** and **c_comp** . The algorithm given in Fig. 3 will perform the appropriate operations.

Vectors **ipipe** and **ipointer** are functionally related by pointer (corre-

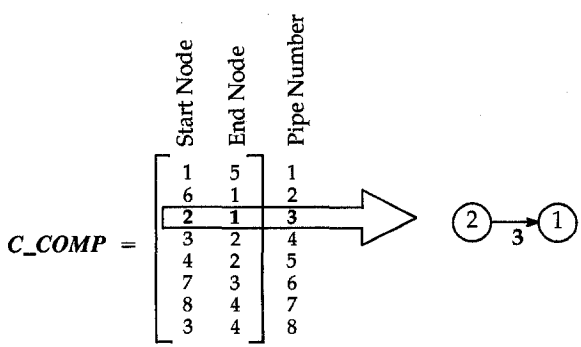

FIG. 2. Matrix c_comp

J. Water Resour. Plann. Manage. 1994.120:116-120.

```
loop i=1,2
        loop j =1 , n_pipe
                k = -j
                if(i.eq.1) k = j
                n = c_comp(j,i)
                ipointer(n+1) = ipointer(n+1) - 1
                ipipe((ipointer(n+1)) = k
        loop end
loop end
```

FIG. 3. Algorithm to Generate ipointer **and** ipipe

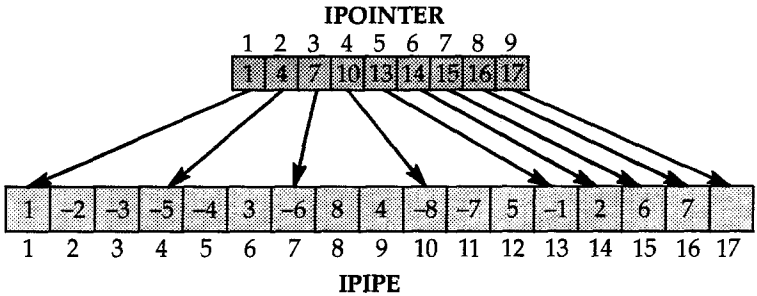

FIG. 4. Matrices ipointer **and** ipipe

sponding) numbers, as illustrated in Fig. 4. Vector **ipointer** contains node numbers, and vector **ipipe** contains pipe numbers. Given a node, these two vectors can return the pipes connected to that node and the direction of flow.

For example, what pipes are connected to node 4? Enter **ipointer** with four, which is paired to the pointer number 10. The first pipe connected to node 4 is pipe 8 (paired with the pointer number 10 in the **ipipe** vector). The pipe is found by entering **ipointer** with five, the next node (4 + 1) that is paired to the pointer number 13. However, the last pipe is one less than that, so the pointer number paired with the last pipe is 12 (found by subtracting one from 13). Consequently, pipe 5 is paired with the pointer number 12 and is the last pipe. Going back from pointer number 12, we find another pipe paired with the pointer number 11, which is pipe −7 (the negative number indicates that water is flowing into the node), see Fig. 1 and (4).

Using these vectors, one can quickly and efficiently determine which pipes are connected to a particular junction node. For example, if we want to determine the pipes that are connected to node 4, we need to do the following:

$$ifirst = \textbf{ipointer}\ (4) = 10 \quad (5a)$$

$$ilast = \textbf{ipointer}\ (5) - 1 = 13 - 1 = 12 \quad (5b)$$

$$\text{loop on } i = ifirst, ilast \quad (5c)$$

$$\text{1st pipe} = \textbf{ipipe}\ (10) = -8 \quad (5d)$$

$$\text{2nd pipe} = \textbf{ipipe}\ (11) = -7 \quad (5e)$$

J. Water Resour. Plann. Manage. 1994.120:116-120.

3rd pipe = **ipipe** (12) = 5 ..... (5*f*)

The negative numbers represent flow into the junction node.

SUMMARY AND CONCLUSIONS

We have presented an efficient method of data processing to supply the necessary inputs to an optimization model. Constructing a **c_comp** matrix and the two vectors, **ipointer** and **ipipe**, greatly reduce the storage requirements and the data-access time. This should make feasible the on-line control of a large water-distribution system utilizing an optimization model, such as MINOS.

ACKNOWLEDGMENTS

This project was supported, in part, by Memphis Light, Gas, and Water.

APPENDIX. REFERENCES

Murtagh, B. A., and Saunders, M. A. (1983). "MINOS 5.0 user's guide." *Technical report sol 83-20*, Systems of Optimization Lab., Dept. of Operations Res., Stanford Univ., Stanford, Calif.

Wood, D. J. (1988). *KYPIPE user manual, computer analysis of flow in pipe networks including extended period simulations*. Coll. of Engrg., Univ. of Kentucky, Lexington, Ky.

J. Water Resour. Plann. Manage. 1994.120:116-120.