AN EVOLUTION PROGRAM FOR PRESSURE REGULATION IN WATER DISTRIBUTION NETWORKS

Dragan A. Savic and Godfrey A. Walters

D.Savic@exeter.ac.uk
G.A.Walters@exeter.ac.uk

Report Number: 94/15

July 25, 1994

Centre for Systems and Control Engineering,
University of Exeter,
North Park Road,
Exeter, EX4 4QF,
Devon,
United Kingdom.

This work was funded by SERC Grant GR/J09796

ABSTRACT

Leakage losses in a water distribution network increase significantly for higher pressures and an obvious way of reducing losses is by reducing network pressures. This report presents a methodology for pressure regulation in a water distribution network using an evolution program, encompassing the principles of evolutionary design and genetic algorithms. The optimisation problem of minimising the pressure heads is formulated with the settings of isolating valves as decision variables and minimum allowable pressures as constraints. The algorithm developed incorporates a steady-state network analysis model based on the linear theory method. Computational results for two example networks demonstrating the effectiveness of the methodology are presented.

KEY WORDS: water distribution networks, pressure regulation, optimisation, evolutionary design, genetic algorithms.

TABLE OF CONTENTS

INTRODUCTION

Leakage Control and Pressure Regulation

Water companies have to maintain a high standard of leakage control since water lost from a distribution network due to leakage can represent a significant economic penalty. Effective leakage control can bring about a substantial saving in energy and treatment costs as well as financial benefit obtained by deferring capital expenditure on the expansion of existing supply schemes and major source development.[1, 2]

In a given network, leakage losses generally increase significantly for higher pressures and an obvious way of reducing losses is by reducing network pressures. However, it is necessary to maintain sufficient pressures (minimum allowable nodal heads) throughout the network to ensure that consumer demands are fully provided at all times. There are other factors which can affect leakage but pressure is the only one that can be easily controlled once the pipelines have been laid. It has been suggested that not only the rate of leakage through a given fracture in a pipeline can be decreased by pressure reduction but also the water use through open ends at the consumers' premises[3] is reduced. Pressure in a network can be regulated by a variety of pressure-control elements, such as valves, booster pumps and fixed head tanks. The problems of optimising control valve settings or choosing the types of pressure-control elements and locating them to minimise excess pressure in a water supply network have been addressed in the past.[4,5,6] In these approaches a limited number of pressure-control elements located within the network were optimised for the purpose of leakage reduction. For the model presented here, pressure regulation is achieved through the settings of isolating valves (i.e., valves which operate fully open or closed and function as devices for shutting off the flow of water through pipes). As such valves will normally be present in all links of

a network, implementation of the pressure regulation regime has little associated cost for considerable potential benefit.

The report presents a methodology for selection of the open/closed settings of all valves in a network to minimise pressures, using an evolution program (EP), a term devised by Michalewicz[7] to encompass evolutionary design and genetic algorithm principles. The optimisation problem is first defined, followed by discussion on its complexity. The evolution program approach is then introduced, with details of the techniques used for its effective implementation. A computer program for hydraulic analysis of water distribution networks specifically developed to be used with an EP for pressure regulation is also presented. The hydraulic solver, which is based on the linear theory method (LTM), enables rapid evaluation of a network for different topological conditions, uses a variable number of LTM iterations and easily accepts changes in input data as required by the EP. Two examples are included to illustrate the practicality of the method, following which findings are discussed and suggestions made for further developments.

Problem Statement

A water supply network, which consists of a number of interconnected pipes and which is fed from one or more reservoirs, can be modelled as a system of arcs interconnected at nodal points. The requirements are expressed by specific demands, both in terms of consumption and head, which are assigned to the nodes of the network. Various mathematical models can be used for the hydraulic analysis of water distribution networks and their accuracy and robustness is well investigated.[8] In this report, usual information requirements for steady-state flow and pressure computations, comprising network topology, reservoir elevations, pipe characteristics (length, diameter, friction coefficient),

ground elevations and demands at each node, are assumed to be known. Secondly, only one loading case, assuming a fixed demand by each customer and fixed reservoir levels, is considered, although in principle several representative demand patterns could be used.

As sections of the network may have to be isolated to repair leaks or for other purposes, isolating valves are frequently installed at specific intervals, the spacing being a function of economics and operating problems. Closing one or more of these valves changes the network configuration and hence the distribution of flows and pressures in the whole network. If nodal requirements stay unchanged, the problem can be posed as: find the optimal settings of all isolating valves (i.e., open or closed) to attain the best possible pressure distribution without compromising network performance (i.e., required flow is supplied to each node and minimum head requirement is satisfied).

Objective Function and Constraints

The objective function for the above stated optimisation problem of valve settings for pressure regulation is given in the form

$$\min_{CV \subseteq V} J = \sum_{i=1}^{N} \left| H_i - H_i^{\min} \right| \qquad (1)$$

where V is a set of all valves in the network, CV is a set of closed valves, N is the number of nodes in the network, H_i is the head at node i and H_i^{min} is the minimum required head at the same node. The minimum head constraint is given as

$$H_i \geq H_i^{\min}; \quad i = 1, \ldots, N \qquad (2)$$

The constraint can be applied to all the nodes or to some critical nodes chosen by close examination of the topography and load pattern of the system.[5] In the current work pressure heads are checked at every node of the network without an increase in complexity of the problem or increase in computer time.

Since solutions infeasible with regards to Eq.(2) may still be useful in the search they are not simply discarded. Instead, the constrained problem is transformed into an unconstrained problem by associating a penalty with constraint violation. The non-negative penalty function is given as

$$P = \sum_{i \in I_v} \alpha \cdot (H_i^{\min} - H_i) \tag{3}$$

where I_v is the set of nodes for which the minimum head constraint is violated and α is a positive penalty multiplier. The objective function takes a new compounded form

$$\min_{CV \subseteq V} J = P + \sum_{i \in I} (H_i - H_i^{\min}) \tag{4}$$

where I is the set of nodes for which the minimum head constraint is satisfied.

Problem Complexity

The above stated optimisation problem is conceptually simple, and by using one of the methods for the hydraulic analysis of water distribution networks, the objective function (J) for any feasible network configuration can be determined. However, the problem is difficult to solve, because of the extremely large number of discrete candidate solutions from which the optimum is to be found. Enormous numbers of local minima and

maxima are also encountered in this type of combinatorial problem, which prevent hill-climbing and other related methods from being effective search techniques.

It is difficult to calculate the size of the solution space, i.e., the total number of feasible networks. A feasible network is one in which all nodes are connected and in which there are no pressure violations. Considering just the connectivity, there is an upper limit of 2^P configurations for a network of N nodes and P pipes, assuming one valve along each pipe which can independently be open or closed. For connectivity, a tree is the minimum permissible network, with N-1 open arcs. Hence the maximum number of valves that can be closed is P-N+1. A lower limit on the number of connected network configurations can be found by calculating the number of tree networks that can be found, using the graph theory method described by Trent.[9] The trees can be enumerated by calculating the value of an (N-1)*(N-1) determinant, where N is the number of nodes in the graph. For a simple base network of N=4 nodes and P=5 pipes in Figure 1, the number of possible tree-like networks is 8, while by enumerating all the possibilities it can be shown that the total number of feasible network configurations (with up to 2 closed valves) is 14. The increase in complexity associated with the increase in network size is illustrated in Table 1 which gives the number of distinct tree-like solution candidates for several example networks.[10 11]

THE EVOLUTION PROGRAM ALGORITHM

Evolution Programs in Pipe Optimisation

The term Evolution Programs (EP) is used here to incorporate the classical Genetic Algorithms[12] (GA) and other Evolutionary Design[13] (ED) methodologies. EP's are general artificial-evolution search methods based on natural selection and the mechanisms of population genetics. These algorithms are best suited to solving combinatorial optimisation problems that cannot be solved using more conventional operational research methods. Thus, they are often applied to large, complex problems that are non-linear with multiple local optima. After reported successes in some problem domains, pipe network optimisation has started to benefit from the use of EP.[11, 14,15,16] The effectiveness of an EP applied to pipe network optimisation problems depends on many factors, but primarily on problem characteristics and EP structure and parameters. The problem characteristics, as described by the preceding problem statement and problem complexity, are well-suited for EP due to the large number of candidate solutions to be examined, non-linearity of the problem and discrete decision space. The choice of EP structure and parameters involves the following decisions:

(1) The most appropriate form of coding or data structure to define the design variables.

(2) The means for generating an initial population of feasible solutions.

(3) The population size to adopt.

(4) The form of fitness function to be used.

(5) The way in which 'parents' are chosen and 'children' generated, including decisions on preferential selection, crossover and mutation.

(6) The way in which the members of a population are replaced.

(7) Criteria for termination of the process.

The present report presents an EP algorithm whose originality is associated with the form of genetic operators and the formation of feasible network solutions. In addition, the ability of an EP to converge to near-optimal solutions under noisy conditions is used to advantage by a rapid hydraulic analysis solver whose initially low accuracy is increased during an EP run.

The Form of Coding

In conventional GA, candidate solutions (in this case network configurations generated from the network base graph) would be represented by strings of numbers (chromosomes) using a binary or non-binary alphabet. Most of the GA devised for pipe network optimisation problems use such coding. However, string coding and the conventional crossover and mutation processes for the problem of selecting a tree network from a non-directed base graph will often generate infeasible solutions. In the paper by Walters and Smith,[11] an alternative data structure based on sets is used to represent decision variables (pipes) within an EP which ensures generation of feasible solutions.

The present algorithm uses a similar concept for representing candidate solutions. A set of integer numbers is used to represent pipes which supply water to the network nodes. A single set (chromosome) can completely define which arcs are closed (no-flow pipes) and which are not. For example, a simple network whose base graph consists of 10 pipes (Figure 2a), with known connectivity, can be represented as in Figure 2b by:

$$A = [2, 3, 5, 6, 7, 8, 9, 10]$$

where the complement set $\underline{A}$ = [1, 4] represents no-flow pipes, i.e., closed valves. Furthermore, the set A is divided into two subsets, one consisting of the arcs that constitute a spanning tree, A_t, and the other which contains the remainder of the set (i.e., a co-tree), A_{ct}. An example of the possible tree and co-tree sets for the network in Figure 2b can be given as (Figure 2c):

$$A_t = [2, 3, 5, 6, 7, 8, 10] \text{ and } A_{ct} = [9]$$

This additional division of the set A is adopted to ensure creation of feasible network layouts from the network base graph and also to set up the equations for hydraulic analysis. The order in which arc numbers are given in the set is not important and Pascal SET type variables and operators are particularly useful for economical storage and easy manipulation of data.

Generation of Initial Population

The initial population consists of feasible solutions generated in a random fashion which will ensure that this population and early generations are likely to contain diverse genetic information. The search continues by moving to a new population in the solution space. The required population size of the conventional (binary) GA applied to pipeline optimisation is discussed by Simpson and Goldberg.[17] However, non-binary coding coupled with the use of the tree-growing algorithm seems to require smaller population sizes to ensure near-optimal solutions.

Generating feasible solutions from a base graph such that all nodes are connected (i.e., a solution that is at least a tree, but which may also contain loops), is not an

easy task. The approach should be hierarchical, i.e., on the first level it should provide easy creation of spanning trees and on the second level it should allow for loop existence in the network. Creation of feasible solutions in the current algorithm is therefore divided into two steps: 1) creation of a spanning tree from the undirected graph, and 2) selection of a number of co-tree members to add to the solution and so create loops.

The first step is implemented through a depth-first-search algorithm which 'grows' a tree from a node chosen at random or from the 'root' (network source). It is expected that generating a tree from a randomly chosen node results in an unbiased tree, while generating it from the root creates trees with characteristics of sensible engineered networks.[11] From this starting node, the algorithm proceeds onto one of the adjacent nodes which has not yet been connected. The arc a_i connecting the two nodes is added to the tree set, $a_i \rightarrow A_t$. The search terminates when all nodes are connected and the tree set then contains the arcs of the spanning tree. If a node is visited before the terminating condition is reached, and there are no adjacent nodes not connected to the growing tree, the algorithm returns to the node previously visited and continues the search for an adjacent, not visited node. The arcs not contained in the tree set are stored in the co-tree set and will be used in the second step for generating candidate solutions (for more on the tree-generating algorithm can be found in ???). Figure 3a shows a looped graph within which a subset of arcs (shown solid) comprise a tree B_t:

$$B_t = [1, 2, 4, 5, 6, 7, 10, 13, 14, 16, 17, 19]$$

The broken lines represent links that connect branches of the tree, i.e., the co-tree B_{ct}:

$$B_{ct} = [3, 8, 9, 11, 12, 15, 18, 20, 21]$$

Creation of the initial population continues through the second step which involves selection of the arcs to be closed. The selection should not be allowed to cause infeasibility of the solution. Feasibility is ensured by preserving the tree set B_t, and therefore withdrawing arcs only from the co-tree set. The selection of the arcs to be withdrawn is done on a random basis with both events, i.e., the arc staying and the arc leaving the co-tree set, having the same probability. The resulting network may look as in Figure 3b, the reduced co-tree set being:

$$B_{ct} = [8, 12, 18, 20, 21]$$

It is worth noting that each of the arcs in B_{ct}, if added to the tree, forms a unique closed loop. Therefore, the number of loops in a given graph is equal to the number of arcs in the co-tree set of the graph.

Hydraulic Analysis

In general, two groups of methods are available for hydraulic analysis of water distribution networks, namely methods based on loop equations (expressed in terms of the flow in each pipe) and methods based on node equations (expressed in terms of heads at junction nodes). It would seem appropriate to use one of the algorithms for solving the node equations when attempting to optimise network operation for pressure regulation. Unfortunately, the thorough analysis of the reliability of various algorithms done by Wood and Funk[8] revealed that methods based on node equations are less reliable than methods based on loop equations. The most promising algorithms based on loop equations are the Newton-Raphson and the linear theory method (LTM). The convergence

rate is comparable for both methods but LTM does not require an initial starting point which is an advantage if many topologically different networks are to be evaluated. Having this in mind, LTM based on loop equations is implemented in the present algorithm.

HYDRAULIC SOLVER

The hydraulic solver will be introduced through the mathematical formulation of the governing equations. The network topology analysis is then presented, with details of the algorithms for creating a spanning tree and identifying loops. Finally, some accuracy considerations relevant to the EP developed are discussed.

Mathematical Formulation

The following mathematical statement of the problem is presented for a general water distribution network consisting of N nodes and P pipes. For each junction node except for the source(s), i.e., $n = 1,2,...,N\text{-}S$, a continuity equation is written as:

$$\sum Q_{in} - \sum Q_{out} = Q_e \tag{6}$$

where S is the number of sources (fixed-head nodes), Q_{in} is the flow into the junction, Q_{out} is the flow out of the junction and Q_e represents the external inflow or demand at the junction node. Under this convention demands Q_e which extract flow from the junction are positive. The next step is to formulate the energy conservation (loop) equations.

For each of the loops in the network, i.e., $l = 1,2,...,P\text{-}N+1$, the energy conservation equation can be written as:

$$\sum h_L - \sum E_p = 0 \tag{7}$$

where h_L is the energy loss in each pipe and E_p is the energy put into the liquid by a pump. If more than one source node is available then an additional energy conservation

equation can be written for paths between any two of the nodes (pseudo loops). This equation takes the following form:

$$\sum h_L - \sum E_p = \Delta E \tag{8}$$

where ΔE is the difference in pressure head between the two source nodes. The head loss h_L to friction associated with flow through a pipe can be expressed in a general form as:

$$h_L = c Q^n \tag{9}$$

where c and n depend on the flow resistance law selected. A similar relationship can be written for the energy loss due to fittings, valves, meters and other network components (n = 2). Although other enhancements of the LTM, such as including pumps and/or pressure reduction valves, can be found in the literature[18] the model developed handles only fixed-head nodes.

Linear theory transforms the non-linear loop equations into linear equations by approximating the head in each pipe by

$$h_L = c Q_i^{n-1} Q \tag{10}$$

where Q_i is an arbitrary initial flowrate. The system consisting of linear equations (6) and linearized equations (8) is solved for flows Q and the new set of flows Q_i is used to linearize the equations (8) and a second solution is obtained. The process continues until

a certain prespecified accuracy level is reached, i.e., until the change of flow between two iterations falls below a threshold value. Wood and Funk[8] tested LTM and four other methods on over 50 actual or proposed network systems in the USA. Twenty-one of those networks contained more than 100 pipes (P_{min} = 117, P_{max} = 509, and $P_{aver\text{-}age}$ = 212.6). The situations included in their study represent a wide variety of systems, some of which incorporate features that increase convergence difficulties. The specific LTM convergence criterion employed was

$$\varepsilon = \frac{\sum |Q - Q_i|}{\sum |Q|} \leq 0.005 \qquad (11)$$

This criterion roughly states that when the average change in flowrates between successive trials is less than 0.5% the calculations cease. The average number of iterations of LTM necessary to reach the required convergence criterion for systems with more than 100 pipes was $ITER_{avg}$= 6.4 ($ITER_{max}$ = 10 and $ITER_{min}$ = 3). In their study, the linear method was capable of meeting the requirements for an accurate solution for every situation investigated. However, convergence problems were encountered with the other methods and even the Newton-Raphson method failed for one of the cases investigated. Other methods were considerably less reliable even for smaller networks. For example, the well-known Hardy-Cross method failed to produce an accurate solution in the majority of situations (51 of 60).

Network connectivity and other usual information requirements for steady-state flow and pressure computations, comprising reservoir elevations, pipe characteristics (length, diameter, friction coefficient), ground elevations and demands at each node, constitute the input for the solver. Output response covers pipe flows and nodal heads

which completely describe the behaviour of the network under specific boundary conditions of the sources and demand nodes.

Network Topology Analysis

This analysis requires identification of a spanning tree and loops of the network. The rationale for generating a spanning tree for a network is twofold: a) to ensure the connectedness of the network and b) to enable easy identification of loops. The formulation of the energy conservation equations of Eq. (8) requires that fundamental loops and pseudo loops (a sequence of pipes connecting two fixed-head nodes) be identified.

Creating a spanning tree

For a network consisting of N nodes and P pipes there is an upper limit of 2^P configurations, assuming one valve along each pipe which can independently be open or closed. However, the number of topologically feasible networks is smaller since only connected networks are acceptable. A spanning tree represents an extreme case of graph connectivity since removing any arc from a tree will disconnect it. Therefore, the connectedness of a network can be verified by means of generating a spanning tree of a base graph. Since the number of arcs in a tree is N-1, the maximum number of valves that can be closed without disconnecting the network is P-N+1.

To formulate energy equations requires that arcs constituting each loop of the network are identified. To enable easier identification of loops and, thus, formulation of the energy conservation equations for the hydraulic analysis a chromosome (i.e., a set of integer numbers used to represent pipes which supply water to the network nodes) is divided into two subsets. The first one consists of the arcs that constitute a spanning tree, A_t, of the base graph and the other which contains the remainder of the set (i.e., a co-tree), A_{ct}. This division of a chromosome is accomplished by creation of a spanning

tree from the undirected graph. The method is based on a depth-first-search algorithm which ‘grows’ a tree from a node chosen at random or from the ‘root’ (network source). It is expected that generating a tree from a randomly chosen node results in an unbiased tree, while generating it from the root generates trees with characteristics of sensible engineered networks. The main operations of the tree-growing algorithm are:

1. Associate label DFI(n) = 0, n = 1,2,...,N, with each node in the base graph.
2. Set j = 1.
3. Initialise a set of arcs within the growing tree A_t = [ ].
4. Identify the starting node n_s at random or as being the ‘root’ of the base graph.
5. Initialise a set of nodes contained within the growing tree C = [n_s] and set DFI(n_s) = j.
6. Identify $A_a(n_s)$ = [arcs in base graph connected to node n_s].
7. Identify end-points $E(n_s)$ of the arcs in $A_a(n_s)$.
8. Eligible adjacent nodes to n_s are end-points which are not contained within the growing tree, i.e., $A(n_s) = E(n_s) - (E(n_s) \cap C)$.
9. If $A(n_s)$ is an empty set, $A(n_s)$ = [ ], i.e., there is no adjacent node which is not yet in the growing tree, then
    a) DFI(n_s) = 0,
    b) $j = j - 1$,
    c) find a node $n_s' \subset C$ for which DFI(n_s') = j,
    d) $n_s = n_s'$,
    d) go to (6).
10. If $A(n_s)$ is a non-empty set, i.e., $A(n_s) \neq$ [ ], then
    a) $j = j + 1$,

b) associate DFI(n) = j with each node $n \subset$ A(n_s),

c) choose one node n_s' from A(n_s) at random,

d) C = C + [n_s'],

e) $A_t = A_t$ + [arc whose end-points are n_s and n_s'],

f) $n_s = n_s$'.

11. Repeat from (6) until all the nodes have been added to the connected set, i.e., a spanning tree is formed.

Figure 4 shows how the above algorithm is applied to an example base graph. The growing of the tree for this example is shown below.

a) A_t = [ ], C = [1] and A(1) = [2, 3],

b) A_t = [2], C = [1, 3] and A(3) = [2, 4, 5],

c) A_t = [2, 5], C = [1, 3, 4] and A(4) = [2, 5, 6],

d) A_t = [2, 4, 5], C = [1, 2, 3, 4] and A(2) = [ ],

e) A_t = [2, 4, 5, 8], C = [1, 2, 3, 4, 6] and A(4) = [5, 6],

f) A_t = [2, 4, 5, 8, 9], C = [1, 2, 3, 4, 5, 6] and A_{ct} = [1, 3, 6, 7].

The arcs contained in the base graph but not picked up by the tree-growing procedure are stored in the co-tree set A_{ct}. This concludes the first step in creating feasible solutions, i.e., growing of a spanning tree from the undirected graph.

Loop-defining algorithm

Before proceeding with the algorithm, it is worth noting that each of the arcs in A_{ct}, if added to the tree, forms a unique closed loop. Therefore, the number of loops (also

called fundamental loops) in a given graph is equal to the number of arcs in the co-tree set of the graph. The algorithm developed is based on the breath-first search method which when applied to each of the arcs in the co-tree set identifies the corresponding loops. Basic steps of the loop-defining algorithm are outlined below:

1. Associate label BFI(n) = 0, n = 1,2,...,N, with each node in the base graph.
2. Set j = 1.
3. Choose one of the arcs from the co-tree set $a_i \in A_{ct}$.
4. Initialise a set of arcs within the loop L = [a_i].
5. Identify the end points n_i and n_i' of the co-tree arc a_i.
6. Initialise a set of nodes contained within a queue Q = [n_i] and set BFI(n_i) = j.
7. Identify A_a = [arcs in the tree set A_t connected to nodes in the queue Q].
8. Identify end-points E of the arcs in A_a which are not labeled, i.e., BFI(n) = 0 for all $n \in$ E.
9. Set Q = [ ].
10. a) set $j = j + 1$,
    b) associate BFI(n) = j with each node $n \in$ E,
    c) set Q = E.
11. Repeat from (7) until n_i' acquires a label.
12. Retrace back from n_i' to n_i (using the tree arcs A_t) in decreasing order of BFI. The arcs belonging to the loop are stored in L.

An example of the loop-defining algorithm applied to the tree defined in Figure 4 and the arc a_i = 6 is given below:

a) $a_i = 6 \in A_{ct}$, L = [6], $n_i = 5$, $n_i' = 3$, Q = [5] and BFI(5) = 1,

b) A_a = [9], E = [6] and BFI(6) = 2,

c) A_a = [8], E = [4] and BFI(4) = 3,

d) A_a = [4, 5], E = [2, 3], BFI(2) = 4 and BFI(n_i') = BFI(3) = 4,

e) BFI(3) = 4 and BFI(4) = 3 $\Rightarrow$ L = [5, 6],

f) BFI(4) = 3 and BFI(6) = 2 $\Rightarrow$ L = [5, 6, 8],

g) BFI(6) = 2 and BFI(5) = 1 $\Rightarrow$ L = [5, 6, 8, 9].

Similarly, all the loops associated with the co-tree members A_{ct} = [1, 3, 6, 7] can be found using the above outlined algorithm.

Accuracy Considerations

Because of the non-linear nature of the loop equations an iterative procedure is used for solving the system of equations. It is obvious that the number of iterations necessary to reach a solution and the overall precision of a solution depends on the relative flow accuracy requirement of Eq. (11). A solution obtained by using LTM with any number of iterations will satisfy the mass continuity equations exactly, but the extent of unbalanced heads for the energy equations will differ for each of the solutions. In general, the average unbalanced head for the energy equations will decrease with the increase in the number of iterations of LTM.

Since numerous evaluations of network flows and heads for different valve settings have to be performed, an efficient way of increasing the speed of an EP run had to be investigated. An obvious means of accomplishing the increase is to decrease the number of iterations of LTM. However, as was explained earlier, this would mean increase in the unbalanced heads for the energy equations. Incidentally, the EP has the

ability to converge to near-optimal solutions under noisy conditions, i.e., can effectively search the decision space even if the objective function values are imprecise due to measurement errors, erroneous readings or similar problems. The only requirement for the EP using less accurate fitness evaluation is to preserve similar ranking between solutions. In other words, if a set of less precise solutions has a similar ranking order to a set of more precise solutions to the same problem, the initial searching of the decision space can be effectively performed using the less accurate, and thus faster, method. In order to use this ability to advantage, a thorough examination of solution accuracy was performed for different network examples. The following section introduces the notion of the fitness function and presents the results of the examination of solution accuracy.

Fitness Function

The fitness function should return a measure of how good any candidate solution is. Since the network configuration which will minimise the objective function J in Eq. (1) is sought, the fitness of a solution must be a function of J.

Before going further with the finess function, the examination of solution accuracy which was performed for different network examples is presented next. The same network flow problems were solved using different allowable numbers of iterations of the LTM, $ITER_{max}$. An example comparison is given in Table 2. The table presents results obtained from the first 15 generations of an EP run for a network of N=34 nodes and P=47 pipes. Row 1 refers to the best solution from the initial population, and subsequent rows refer to sequential improvements on the best solution, all based on an objective function J calculated with $ITER_{max}$= 3 (column 3). Column 2 gives the generation in which the improvement was identified. Column 4 ranks the solutions, while columns 5 and 6 give the true objective function value (when no upper bound on $ITER_{max}$ is im-

posed but the required accuracy is set to be $\varepsilon = 0.005$) and the new ranking order respectively. It can be observed that the ranking order of solutions obtained by using only 3 LTM iterations differs only slightly from the ranking of "exact" solutions, i.e., solutions that satisfy the criterion of Eq. (11). However, this holds true only for the solutions in the first few generations, which are likely to be diverse and of widely varying objective function values due to effects of introduced randomness. Later in the evolution when solutions converge, the difference between them diminishes and the error introduced due to a limited number of LTM iterations tends to destroy the actual ranking. Therefore, the experiments with variable accuracy justify the idea of using less LTM iterations in the first few generations with an improvement in accuracy later in the evolution.

Because of the relative imprecision of the calculations in the first stages of the evolutionary process, penalties are not considered if the extent of constraint violation is of the same order of magnitude as the error introduced due to the limited number of iterations. Subsequently, stricter penalties are applied to a potential solution when higher accuracy is introduced at later stages of the evolutionary process.

Regardless of whether penalties have been included in the objective function or not, the following relation between the objective function value J_k for the solution k and the corresponding fitness f_k is used

$$f_k = \frac{J_{\max} - J_k}{J_{\max} - J_{\min}} \tag{3}$$

where J_{min} and J_{max} are the best and the worst solution candidates of the population respectively. This linear scaling scheme discriminates between candidates when all members have similar values of objective function.

Generational Replacement

After generating the initial population, the generational replacement scheme is applied next to generate the new population from the old one. The new population then becomes old and the process is repeated. To avoid a possible loss of the best population member the generational replacement scheme is augmented by an elitist strategy in which the best solution from the old generation is kept in the new generation. Creation of members of the new population is performed through the mating (recombining) of a pair of members from the old population. The process starts with *selection of parents* for mating. The second step, *creation of children*, differs from the conventional GA but follows the basic idea by which genetic information of parents is conveyed to the children. This step is then performed by means of new genetic operators.

Selection of parents

A process similar to the roulette wheel random process[19] for the selection of parents is adopted in the current algorithm. Every member of a population is assigned a certain segment of a wheel, the size of the segment being proportional to the fitness of the member. The selection probability P_k, for a particular member k, is thereby obtained as

$$P_k = \frac{f_k}{\sum_{i=1}^{M_{pop}} f_i} \tag{4}$$

where M_{pop} is the number of candidate solutions in the population. The probability of being selected depends on the size of the segment allocated to each of the solutions. A random number generator is used to select two members out of a population on the basis of selection probability defined as in Eq. (9).

Creation of children

Suppose now that for the base graph G in Figure 5a, a population of members is generated as previously explained. After two parents have been selected for mating, e.g., A and B in Figure 5b, two feasible children will be created by applying new genetic operators as follows:

1. Combine the two arc sets and form $A \cup B$ and $A \cap B$ (Figure 5c) which together constitute a pool of genetic information which will be used in the process.
2. Grow a random tree as a subset of $A \cup B$ as previously described in Generation of Initial Population, e.g., $C_t \subset (A \cup B)$ in Figure 5d. It should be noted that there is at least one feasible tree because the union operator cannot infringe on feasibility of the network layout.
3. Assign the members of $A \cap B$ which were not included in C_t to the set C_3 (Figure 5e), or in set notation $C_3 = (A \cap B) - ((A \cap B) \cap C_t)$. The reason for keeping track of all the arcs common to both parents is that they are likely to contribute to the high fitness of the parents and thus should remain in the genetic material conveyed to the child.
4. Form two sets of arcs which were not common to the two parents: a) $C_{4a} = A - (A \cap B)$, and b) $C_{4b} = B - (A \cap B)$. Figure 5f shows the graphs corresponding to the two sets.

5. Combine the two sets and form $C_5 \subset (C_{4a} \cup C_{4b})$ such that each member of either of the sets has a 50% chance of being included in C_5. An example of a set C_5 is given in Figure 5g. The probability threshold for inclusion/exclusion of an arc is adopted simply because only one of the parents has the arc and, thus, there is a 50% chance it will remain in the genetic material of a child.

It should be noted that up to this point the pool of genetic information contains only the arcs found in one or both of the parents. It is then likely that the operations 1-5 tend to narrow the search, i.e., do not consider arcs excluded from both parents, i.e., the set of arcs $G - (A \cup B)$. To safeguard against early loss of valuable genetic material two operators similar to bit mutation in classical GA are introduced.

6. Form a set of arcs $C_6 \subset G - (A \cup B)$, such that a very small probability is associated with an arc being allowed to remain in C_6 (e.g., $P_{mut}^{+} = 0.01$). If this probability, which will be called the positive mutation rate, is set too high, the process will degenerate into a random search.
7. Form a set of arcs $C_7 \subset C_3 \cup C_5$, such that, again, a very small probability is associated with an arc being allowed to remain in C_7. Similarly to creation of C_6, if this probability, which will be called the negative mutation rate, is set too high, the process will degenerate into a search among tree-like network structures.
8. Form a co-tree set $C_{ct} = C_3 + C_5 + C_6 - C_7$.

The two sets C_t and C_{ct} define one child of A and B. The other child is created by repeating the steps 5-7. A different approach to step 5, which will allow creation of both children in just one pass through steps 1-7, is also possible. This approach would permit one child to have the arcs C_{4a} of parent A and the other child to have the arcs C_{4b} of par-

ent B. Either method creates two children which have feasible layouts and are based strongly on the genetic information of their parents.

NUMERICAL EXAMPLES

A program based on the methodology presented has been written in Pascal and implemented on an IBM compatible computer. It has been used to test a number of different networks of different sizes and all computations were performed on a PC with a 486DX processor. The performance of the method is illustrated using two sample problems.

The layout of the first, a single source network with N=9 nodes and P=17 pipes, is shown in Figure 6. The input data for this network is summarised in Tables 3 and 4. The source is a node with a fixed head of 60 m. The minimum acceptable pressure requirements for the remaining 8 nodes are defined as 10 m above ground level. The problem is chosen to be small enough to allow for a complete enumeration and thus provide a 'true' optimum which can be compared to that solution obtained by the EP developed.

The choice of parameters for standard GA problems can have a significant impact on performance of the algorithm. Many studies have been devoted to finding the best possible combination of the parameters among which the one by Grefenstette[20] even suggested using a meta-genetic algorithm for finding values of the GA parameters. The significance of the parameter values for this study were investigated by repeatedly running the program with different population sizes and positive and negative mutation rates. In addition, the number of generations using the lower accuracy of hydraulic analysis has been varied. It was found that relatively small population sizes (N_{pop}= 10 - 20) and mutation rates of p_{mut} = 0.10 - 0.20 give the best results. As far as accuracy of the hydraulic analysis is concerned, the best solutions were found if the first 100 - 200 generations are evaluated using a limited number of iterations of the linear theory method.

The total number of possible combinations of valve settings for this network, the upper bound, is 2^{17} = 131,072. The lower bound on the number of feasible solutions is calculated as the total number of trees that can be formed from the network base graph and is 5,716. A computer program was written to perform a complete enumeration of the problem space. The number of feasible networks found was 46,624 (35.6% of the total number of combinations) from which the optimum was found to be J = 46.8 m, with 7 valves closed. The valves identified are located on pipes: 4, 5, 7, 8, 10, 13 and 16.

By repeatedly running the EP, the same optimum value was found. Figure 7 shows rapid convergence to a near-optimal solution which is achieved in the first 100 generations. A jump in the plot occurs when the more accurate analysis is introduced and further progression to the optimal solution is less rapid and requires many generations. As many as 1000 generations may be necessary to achieve a solution within 2.7% of the true optimum.

The second example network, whose layout is shown in Figure 8, consists of N=34 nodes and P=47 pipes. Table 5 shows the demand and ground level values for all the nodes, which are supplied by one source at a head of 80 m. The minimum required pressure head at all nodes is set to 15 m. Relevant pipe data are given in Table 6. In this example there are 1.4×10^{14} (2^{47})possible candidate solutions and a complete enumaration is impracticable. However, by evaluating a large number of solutions created by random valve closures it was found that, on average, only 1 in 10,000 generated solutions is connected. To assess the reduction in excess pressures, the pressure distribution is first obtained for the case where there are no closed valves in the network. The objective function value for this case of no pressure control is J = 1257.5 m. By repeatedly running the EP with the same parameters as in the first example, the lowest objective

function value was found to be $J = 473.7$ m. Figure 9 shows a plot of the objective function value for the best solution found against the number of generations for a particular run. The best solution identifies closure of 9 valves. These valves are located on pipes: 4, 14, 20, 30, 34, 36, 44, 45 and 46. In contrast, an alternative, near-optimal solution, having an objective function value of $J = 476.2$ m, identifies a very different set of 10 valves CV = [3, 4, 17, 21, 25, 29, 32, 33, 34, 43]. If a smaller number of valves is more desirable, then a solution which requires only 5 valves to be closed, i.e., CV = [23, 33, 34, 36, 43] can be used. The objective function value for this solution ($J = 494.3$ m) is only 4.3% greater than that for the best solution found. In addition, if solutions which are infeasible with respect to the minimum head requirement are analysed, those with relatively small minimum head violations my be considered acceptable in some circumstances. For example one such solution is to close [4, 7, 26, 37, 47]. This gives an objective function value of $J = 371.0$ m, an improvement of 21.7% over the best solution identified but with a minimum head violation of 2.0 m at node 31. It is also important to note that the EP was allowed 50,000 evaluations (number of generations $\times$ number of members in a population) per run which represents only 0.00036% of the expected number of feasible (connected) networks.

DISCUSSION AND CONCLUSIONS

The variations of pressure in a network are known to affect water losses significantly. While minimum pressure is required to be maintained throughout the network, the excess pressures are responsible for increased leakage and probability of pipe failure. In the work described in this report, the problem of excess pressure minimisation by optimising the settings of isolating valves has been presented. The reason for considering isolating valves for pressure regulation is that they are present in most of the networks and, thus, pressure reduction may be achieved at low cost without investing in new network elements.

The evolution programming approach based on natural selection and the mechanisms of population genetics was used to develop an efficient optimisation algorithm. The algorithm is coupled with an hydraulic analysis program which incorporates fully the governing equations of the network. Since numerous hydraulic analyses need to be performed the use of a limited number of iterations of LTM in the first stages of the evolutionary process considerably accelerates the search. The program developed ensures that only feasible network layouts are used in the search by preventing any of the nodes being isolated from the rest of the network which may otherwise happen due to the closure of one or more valves. It also allows solutions infeasible with respect to the minimum head requirement to stay and help guide the search. The program developed runs on a personal computer and efficiently provides near optimal solutions.

The results show that the program identifies not just a single solution, but a range of near-optimal solutions which may be radically different in terms of the set of selected decision variables. For example, two solutions may have similar objective function values, achieved by closing an almost completely different set of isolating valves indicating a solution space with a large number of local optima. A solution which

satisfies some additional criterion (e.g., less valves involved, most of the identified valves remotely controlled, etc.) may be more desirable in practice even if its pressure distribution is less than optimal, but not far off the target.

Classical sensitivity analysis may be coupled with a hydraulic simulation program to provide a near-optimal solution. However, when applied to a problem with a highly constrained multi-dimensional feasible space (as it is in this case) classical sensitivity analysis may not be very efficient in finding near-optimal solutions. Even a very skillful user of a hydraulic solver would need many fruitless runs to provide a near-optimal solution. Thus, an additional benefit of using the hydraulic solver within the EP lies in the fact that one may view numerous near-optimal solutions as a form of automated sensitivity analysis.

ACKNOWLEDGEMENT

This work was funded by the Science and Engineering Research Council grant GR/J09796.

REFERENCES

[1]**Water Authorities Association**. Leakage control policy and practice. Report originally published as *National Water Council/Department of the Environment Standing Technical Committee* report no. 26, NWC, London, 1974.

[2]**Setford, M.T**. An economic appraisal of active and passive methods of leakage control, *J. Inst. Water Engrs. Scientists*, 1985, **39**, 6, 494-500.

[3]**Bessey, G.** Some developments in pressure reduction, *J. Inst. Water Engrs. Scientists*, 1985, **39**, 6, 501-506.

[4]**Sterling, M.J.H. and Bargiela, A**. Leakage reduction by optimised control of valves in water networks, *Trans. Institute of Measurement and Control*, 1984, **6**, 6, 293-298.

[5]**Germanopoulos, G. and Jowitt, P.W.** Leakage reduction by excess pressure minimisation in a water supply network, *Proc. Instn Civ. Engrs*, Part 2, 1989, **87**, 195-214.

[6]**Hindi, K.S. and Hamam, Y.M**. Locating pressure control elements for leakage minimization in water supply networks: an optimisation model, *Eng. Opt*., 1991, **17**, 281-291.

[7]**Michalewicz, Z.** Genetic Algorithms + Data Structures = Evolutionary Programs, Springer-Verlag, 1992.

[8]**Wood, D.J. and Funk, J.E**. Hydraulic analysis of water distribution systems, *Water Supply Systems, State of the art and future trends*, E. Cabrera and F. Martinez (eds.), Computational Mechanics Publications, 1993, 41-85.

[9]**Trent, H.M**., A note on the enumeration and listing of all possible trees in a connected linear graph, Proceedings National Academy of Sciences, 1954, **40**, 10, 1004-1007.

[10]**Lohbeck, T.K.** Genetic Algorithms in Layout Selection for Tree-like Pipe Networks, M.Phil. Thesis, University of Exeter, Great Britain, 1993.

[11]**Walters, G.A. and Smith, D.K.**, Evolutionary design algorithm for optimal layout of tree networks, ASCE Journal of Computing in Civil Engineering (to be published).

[12]**Holland H. J**., Adaptation in natural and artificial systems, University of Michigan Press, Ann Arbor, 1975.

[13]**Rechenberg, I**., Evolutionsstrategie, Frommann-Holzboog, Problemata 15, 1973.

[14]**Walters, G.A. and Lohbeck, T.K**., Optimal layout of tree networks using genetic algorithms, *Eng. Opt.*, 1993, **22**, 27-48.

[15]**Cembrowicz, R.G**., Water supply systems optimisation for developing countries, Pipeline Systems, B. Coulbeck and E. Evans (eds.), Kluwer Academic Publishers, 1992, 59-76.

[16]**Murphy, L.J. and Simpson, A.R**., Genetic algorithms in pipe network optimisation, Research Report No. R93, Department of Civil and Environmental Engineering, University of Adelaide, 1992.

[17]**Simpson, A.R. and Goldberg, D.E.**, Pipeline optimisation via genetic algorithms: from theory to practice, Water Pipeline Systems, D.S. Miller (ed.), Mechanical Engineering Publications Ltd., London, 1994, 309-320.

[18]**Jeppson, R.W**., *Analysis of flow in pipe networks*. Ann Arbor Science, Michigan, 1983.

[19]**Davis, L.**, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[20]**Grefenstette, J.J.**, Optimisation of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, 1986, SMC-16, 1, 122-128.