A methodology for least-cost design of invulnerable water distribution networks

Avner Kessler*, Lindell Ormsbee* and Uri Shamir†

Water distribution networks, like other engineering systems, are subjected to random failures. The failures may occur in the elements of the network (mostly pipes) or in the nodes (pipe junctions and demand nodes). Since a breakdown in water supply is not acceptable for more than a few hours, the network should be invulnerable to some extent, that is, it must include several independent alternatives of water supply to every consumer. Adopting an invulnerability degree of two alternative supplies, a new methodology is presented for the least-cost design of water distribution networks. According to this methodology the design process is controlled by the designer via three distinct stages. In the first stage the alternative paths are allocated using graph theory algorithms. In the second stage the minimum hydraulic capacity (diameters) of each path is determined using a linear programming model, and in the third stage the solution is tested by a network solver for various demand patterns. An example for the first stage and its relevant graph theory algorithms are included.

Keywords: water supply, networks, reliability, optimal design

Urban water distribution networks are designed to supply water for domestic and industrial purposes 24 hours a day. A breakdown in the water supply for more than a few hours has a severe undesirable effect on the consumers' activity. Although the damage involved might be measured by its cost, it is usually agreed that such events are unacceptable in a well developed community. Since a random failure of a component is unavoidable, water distribution networks should be invulnerable to some extent to the impact of failure. That is, adequate water supply should be maintained under a random failure of one or more of its elements or nodes. It is the purpose of this paper to develop and evaluate a methodology for the design of invulnerable least-cost water distribution networks.

System invulnerability may be measured by the number of independent alternatives which exist to perform a particular job. In water distribution networks, system invulnerability is measured by the number of distinct paths (series of pipelines) between the source and every demand node (consumer). A path is counted only if it is of an adequate hydraulic capacity, i.e. it provides a minimum pressure and demand flow at the demand node. Note that the design of invulnerable water distribution networks is more complicated than other common networks. For computer and telephone networks the path capacity is automatically fulfilled when a path exists, while for electrical networks the cost of the path is almost independent of its capacity. The solution of water networks is further complicated by the nonlinear terminal equation of the hydraulic elements (flow-headloss relationship along a pipe, valve and pump).

During the last 20 years much research activity has been directed toward development of least-cost solutions. That is, given the layout of the network and a number of demand patterns, what is the network capacity (pipe diameter, pump size) which results in minimum cost. Unfortunately, the minimum cost objective by its nature opposes any network redundancy. It is well known that a least-cost solution approaches a tree shaped configuration[1]. Such a configuration represents the most weakly connected network, i.e. any single pipe failure will disconnect some of the consumers from the source node.

Attempts to incorporate invulnerability in the design process have traditionally been provided by the requirement for a looped network, i.e. each node (consumer) is linked with two or more edges (pipes). This results in prevention of a tree shaped network. No consideration is given to the best set of loops nor to their minimum capacity needed to overcome a random pipe failure. This approach was implemented by Alperovits and Shamir[2], who constrained the minimum allowable capacity (diameter or flow) of each pipeline. An improvement was presented by Morgan and Goulter[3] where the layout of the network is optimally determined such that every node must be linked by at least two pipes. This method results in a better set of loops, but still fails to consider the proper selection of the loops' capacity.

Goulter and Coals[4] have included the failure probability of the pipe (correlated with pipe diameter and pipe length) to minimize the probability of a node isolation. First, the network is optimally determined subject to a minimum failure probability allowed in each pipe. Second, the probability of node isolation is decreased by minimizing the deviation of the failure probabilities among the pipes which are connected to a particular node.

* Department of Civil Engineering, University of Kentucky, Lexington, KY 40506-0046, USA
† Faculty of Civil Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel

0263-0257/90 $3.00+.12
© 1990 E. & F. N. Spon Ltd

Although this method is superior to previous methods it may only guarantee a local type of reliability (around a node) instead of a global type (between the source and the consumer set).

Rowell and Barnes[5] have developed a unique approach in which the limitation of a local reliability is partially overcome. Their method consists of two stages: (1) The network is solved to obtain a minimum cost solution, thus defining a tree shaped network; (2) The tree network is augmented by a new set of pipes which is optimally selected in the following manner. Each pipe of the tree network is sequentially removed and the corresponding set of isolated nodes is identified. For each set of isolated nodes a minimal set of new pipes is determined by which the network is reconnected. The capacity of each minimal pipe set is then adjusted to meet the demand flow of the isolated set of nodes. The principal weakness of the method is the flow adjustments in the second stage. Such an adjustment does not guarantee hydraulic consistency with regard to the hydraulic gradients and the pressures. Moreover, the second stage is solved using an integer programming, which is not practical for large networks.

A comprehensive solution of a minimum cost problem subject to reliability constraints was recently introduced by Su *et al.*[6]. The method consists of three hierarchical stages which are solved iteratively. The lower stage uses a network solver to evaluate any combination of pipe failure as specified by the intermediate stage. The intermediate stage uses a reliability model which scans all the minimal cut sets (minimum sets of pipe failure which reduce the pressure beyond its minimum level) in order to compute the network reliability. The upper level makes use of a non linear programming (model GRG2) to define the most economical pipe diameters subject to a minimum network reliability. There is no doubt this method gives the best qualitative and quantitative results compared with previous methods. However its practical implementation is severely limited due to the excess computer execution times required. For example, evaluation of a network of 17 pipes and 4 loops required 200.5 minutes of computer time on a CDC Cyber mainframe.

Following the difficulties encountered so far, it seems that an efficient, closed form, optimal solution for invulnerable water distribution networks is still not available. Thus, a new approach is suggested in which the solution is controlled by the designer, using a number of efficient and fast subprograms. Following this approach a least-cost network is found which can withstand a single random failure without loss of service. The failure being considered may be any pipe breakage or a pipe junction failure, depending on the design requirements. Limitation to a single failure is justified by the relatively small probability of having two simultaneous failures within a common repair period (i.e. 1–2 days). Two failures which are separated by a larger period may be considered as two independent single failures. These probabilities are based on the assumption that all failures are independent and may be characterized using a Poisson Process.

The suggested method consists of the following three stages. In the first stage two edge-disjoint paths are allocated between the source and every demand node. These paths are selected by the designer using advanced techniques of graph theory. In the second stage the minimum hydraulic capacity (diameters) of each path is determined by a linear programming model. The linear model is based on the method of Karmeli *et al.*[7], originally used for an optimal design of a tree-shaped network. In the last stage the solution is tested by a network solver for multiple demand patterns. An illustrative example of the first stage and its relevant algorithms are included.

Connectivity measurements

The first condition for an invulnerable network is the existence of two or more distinct paths between the source and every demand node. This property, termed topological invulnerability, is well studied in graph theory with respect to graph connectivity. The terminology of graph theory as it relates to a water network is given in Appendix 1. Consider the water network as an undirected graph, the graph connectivity is defined by the minimum number of elements whose removal from the graph disconnects some nodes from the others. A distinction is made between the type of the removed elements which can be nodes or edges. The following are the various definitions of the connectivity measurements:

Edge connectivity: The number of edges in the minimum cut-set, i.e. the cut-set with fewest number of edges. Equivalently, the edge connectivity is the minimum number of edges whose removal disconnects the graph. A *k-edge-connected graph* is a graph with an edge connectivity of k. A *bridge* is a cut-set of one edge and a graph is *bridge-connected* if it contains no bridges, i.e. $k>1$. Notice that every edge in a tree is a bridge, therefore the tree represents the most weakly connected graph. An important contribution of graph theory towards the evaluation of topological invulnerability is made by Menger's theorem[8], known as the *maximum flow-minimum cut-set* theorem:

> Let $\lambda(i, j)$ be the maximum number of edge-disjoint paths between nodes i and j. The edge connectivity is equal to the minimum $\lambda(i, j)$ for all (i, j).

The immediate implication of Menger's theorem is that in a k edge-connected graph there are at least k paths of disjoint edges between every pair of nodes. In case of a bridge-connected graph it follows that for every two edges there exists a circuit (not necessarily simple) which goes through both.

Node connectivity: The minimum number of nodes whose removal, along with the removal of their incident edges, disconnects the graph. A *k-node-connected graph* is a graph whose node connectivity equals k. A graph is said to be *separable* if it is a 1-node-connected graph. The node whose removal from a separable graph disconnects the graph called a *cut-node.* Menger's theorem is applied to node connectivity in the following way:

> Let $\mu(i, j)$ be the maximum number of node-disjoint paths between nodes i and j. The node connectivity is equal to the minimum $\mu(i, j)$ for all (i, j).

It follows that for a k-node-connected graph there are at least k paths of disjoint nodes between every pair of nodes. In case of a 2-node-connected graph there exists a simple circuit which goes through every pair of nodes. Cut-nodes and bridges can be efficiently detected by the DFS (Depth First Search) algorithm as outlined in Appendix 2.

Examining the above connectivity measurements it can be seen that edge and node connectivity directly measure the topological invulnerability of a network, that is, the number of alternative paths which exist between the source and every demand node. Edge connectivity is related to the edge disjoint paths while node connectivity is related to the node disjoint paths. Since every set of node-disjoint paths is also a set of edge-disjoint paths (but not the reverse) the node connectivity is always less or equal to the edge connectivity.

While the evaluation of node and edge connectivity might require a special technique, their upper bounds can be computed easily. Considering all the cut-sets by which a node becomes isolated, the number of edges in each cut-set is equal to the node degree of the isolated node. The minimal node degree is therefore greater than or equal to the minimal cut-set of the entire graph, previously defined as the graph edge connectivity. Summarizing the above relationships we have:

$$(\textit{node connectivity}) \leq (\textit{edge connectivity}) \leq (\min \textit{ node degree}) \quad (1)$$

For example, a simple circuit which passes through all nodes (known as a *Hamiltonian graph*) is of a minimum node degree equal to the node and the edge connectivity. Such a graph is the best 2-node and 2-edge-connected graph in view of the minimal number of edges needed to connect a given number of nodes.

Figure 1 shows four different graphs with the same number of nodes. The edges are selected to cover all the possible relationships between: minimum node degree, edge connectivity and node connectivity. *Figure 1a* is a spanning tree with the minimum number of edges. *Figure 1b* is of minimum node degree, equal to 2, but still has a bridge. *Figure 1c* is a 2-edge-connected graph and *Figure 1d* is a 2-node-connected graph. The connectivity measurements of each graph are summarized in *Table 1*. Notice that the optimal 2-connected graph (minimum number of edges) is a simple circuit with 6 edges (see *Figure 1d*) compared with 7 and 8 edges in *Figures 1b* and *c*, respectively.

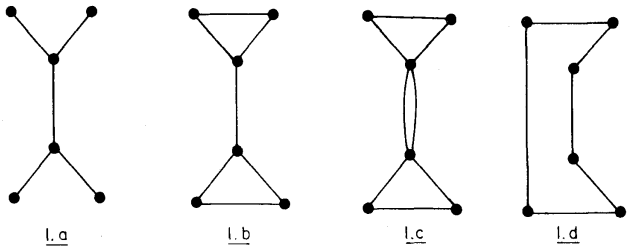

Fig 1 Graphs of various connectivity measurements

Table 1 Connectivity measurements of the graphs in *Figure 1*

Connectivity measurement	1.a	1.b	1.c	1.d
Number of edges	5	7	8	6
Minimum node degree	1	2	2	2
Edge connectivity	1	1	2	2
Node connectivity	1	1	1	2

Methodology development

It was shown in the previous section that a simple circuit has the minimum number of edges by which a 2-node-connected graph is established. The mechanism of this favourable feature is best explained by an illustrative example, shown in *Figure 2*. The network depicted consists of five consumers supplied by a single source. Since there are two node-disjoint paths between the source and each of the consumers, every node failure (or edge failure) will leave one path unattached. The paths in each pair are of opposite direction, therefore it is possible to divide all paths into two subsets according to their direction. Gathering the paths in each subset defines two spanning trees, rooted at the source, as shown in *Figure 2*. Although there is overlap between the two trees, the paths defined by them are node and edge disjoint. Notice that a common edge must be of opposite direction in each tree, otherwise its removal will disconnect all its downstream consumers. One can easily verify that a simple circuit provides the maximum overlapping between the trees. Consequently, the simple circuit required the minimum number of edges to establish a 2-node and a 2-edge-connected graph.

Consider the simple circuit, shown in *Figure 2*, to be a water distribution network. An appropriate hydraulic capacity (diameter) is required for each edge (pipeline). Utilizing the concept of the two trees the appropriate capacity is ensured if and only if each tree can handle the hydraulic requirements by itself (the required flow with an adequate pressure at each node). Notice that unless the pipe failure occurs in the immediate vicinity of the source, a pressure redundancy (above the minimum pressure at each node) exist for all other failures.

The presence of a pressure redundancy is a non-avoidable consequence of the circuit configuration. Its proof stems from the necessary conditions imposed on each overlapping tree in order to fulfil independently the minimum pressure and flow demand in every node. Assuming a failure, not adjacent to the source, each tree delivers only a portion of its design discharge, therefore increasing the pressure in every node. The pressure redundancy will be amplified with an increase of the circuit size (i.e. number of nodes and flow demands). As a consequence, the invulnerable least-cost solution for an actual size network is not necessarily a single circuit (Hamiltonian graph). Other network configurations, usually composed of a number of small circuits, may provide less pressure redundancy and therefore a better least-cost solution.

Although the simple circuit does not provide an optimal solution for an actual water distribution network, the concept of the two overlapping trees is still valid. A general network and its corresponding trees are shown in *Figure 3*. These trees define two disjoint paths between the source and every node, such that each path is included in a different tree. The importance of the trees with respect to the design of an invulnerable network is due to the following two contributions: (1) There exists an alternative path between the source and every node following a random node (or edge) failure. (2) It defines a simple procedure by which the hydraulic capacity of the network

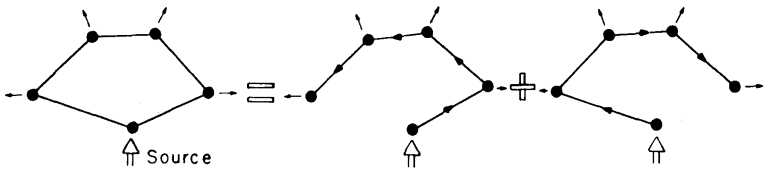

Fig 2 The two trees correspond to a simple circuit

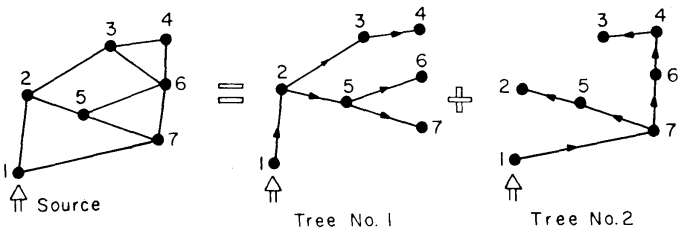

Fig 3 Two overlapping trees in a looped network

is determined (given in the following section). These contributions should be compared with the previous available alternative of path enumeration[6] which is currently impractical due to computation time.

The existence of two spanning trees, as found in *Figure 3*, defines a new connectivity measurement named a *2-tree-connected graph.* Usually, a number of pairs of trees can be identified for a given layout of water distribution network. There is no way to determine the best pair of trees prior to a full hydraulic evaluation of each pair. Nevertheless some guidelines for selecting the two trees may be derived, based on the above discussion:

1. Maximize the number of overlapping edges between the two trees.
2. Minimize the number of overlapping edges in a sequence, i.e. along a common path between the two trees.

The first guideline assumes a correlation between the number of edges and the cost of the network. The second guideline assumes that the pressure redundancy is proportional to the number of the overlapping edges in a sequence, the same as in a single circuit (see *Figure 2*).

The two-tree algorithm

As shown in the previous section, a topological invulnerable network (one able to sustain a single failure) can be constructed by combining two spanning trees. An algorithm and the necessary conditions for the trees' construction are given following the work of Itai and Rodeh[9]. Their work was concerned with the development of a computer protocol to communicate between a number of asynchronous processors over an unreliable network. The algorithm begins with a numbering procedure in which a different number is attached to each node. The numbering procedure, known also as *s-t numbering*, is slightly changed for our special purposes. The definition of the *s-t* numbering used in this paper is as follows:

Let $G(V, E)$ be a 2-node-connected graph of (nn) nodes with 1 source and $(nn-1)$ consumers. (s, t) is an edge in G where s is the source and t is any other node. The function $g: V \rightarrow \{1, 2, \ldots, nn\}$ is an $s-t$ numbering if the following conditions are satisfied:

1. $g(s)=1$ and $g(t)=nn$.
2. Every node v, other than s and t, has two adjacent nodes u and w such that: $g(u)<g(v)<g(w)$.

Lempel *et al.*[10] have shown that every 2-node-connected graph has an $s-t$ numbering. Even and Tarjan[11] gave a linear algorithm, based on DFS (Depth First Search), to find such a numbering. A detailed description of the DFS and the *s-t* numbering algorithms is presented in Appendix 2 and Appendix 3, respectively.

Once a 2-node-connected graph has been numbered according to the *s-t* numbering, each node (not including s and t) is adjacent to at least two nodes whose numbers are lower than and higher than the number of the node itself. It follows that a node can be reached from the source by two disjoint paths, one from the direction of the lower numbers and the second from the direction of the higher numbers. Based upon this property the two trees that satisfy the path-disjoint condition are constructed using the following algorithm:

Tree No. 1: Choose for every node $v \neq s$ an edge (u, v) such that $g(u)<g(v)$. Direct this edge from u to v. The edge (s, t) is not available for node t.
Tree No. 2: Choose for every node $v \neq \{s, t\}$ an edge (w, v) such that $g(w)>g(v)$. Direct this edge from w to v. The edge (s, t) is included and is directed from s to t.

The above algorithm results in two directed spanning trees both of which are rooted in the source. Tree No. 1 is directed from the lower number towards the higher numbers while tree No. 2 is directed in the opposite direction. A sufficient condition for the construction of the two trees is a 2-node-connected graph, which is needed to define an *s-t* numbering. Itai and Rodeh[9] have shown that this condition may be reduced to a 2-edge-connected graph by developing an *s-t* numbering inside the blocks of a separable graph.

The original algorithm[9] can produce several solutions. The set of potential solutions is defined by the number of edges that are incident to every node and that satisfy the relationship between the node numbers. Taking for

example tree No. 2 in *Figure 3*. Both edges, (7, 5) and (6, 5), are possible candidates for node 5. Choosing edge (6, 5) instead of (7, 5) will change the final configuration of Tree No. 2 significantly. No consideration was made in the original algorithm for a particular solution, since the layout configuration has very little impact on the cost of computer networks (for which the method was developed. In water distribution networks, however, the cost is highly dependent on the network configuration. Therefore an optimal selection of the trees is needed.

The best edge in each step of the 2-tree algorithm may be selected based on the judgement of the design engineer or by using the two general heuristic guidelines outlined at the end of the previous section. Theoretically, tree No. 1 becomes a least-cost tree if, for every node, the least-cost edge, satisfying the numbering conditions, is chosen. The cost of the edge is not known *a priori* but can be approximated according to its length and its relative location. An alternative criterion which overcomes the need to approximation is given by the shortest distance tree. This tree defines the shortest path between the source and every other node, using the edge lengths only. Bhave[12] has shown that such a tree tends to approach the optimal solution of a water distribution network with a single source. Such a tree also prevents the unfavourable opportunity of getting a tree with a long path similar to a single simple circuit. Many efficient algorithms are available for the evaluation of the shortest distance tree. The one which is most popular and can be directly applied in our procedure is the Dijkstra algorithm[13].

Tree No. 2 may be evaluated similarly to tree No. 1, except that the edges already belonging to tree No. 1 may be assigned a zero cost. Use of such a zero assignment will force the second tree toward a maximum overlapping between the trees. The designer may use his own judgement to choose the best edge that meets the numbering condition in every stage of the trees' construction. Construction of the trees in this manner would be greatly facilitated using interactive computer graphics. An example of the tree construction is given in the next section.

An illustrative example

The construction of the 2-tree redundant network is illustrated by means of a simple network. The network is of a single source at node s, as shown in *Figure 4*. Applying the DFS algorithm, with node t as a starting node, the tree edges and the back edges are identified. For each node v, where $v \neq t$, the following numbers are designated: the DFS number, $k(v)$, its father, $f(v)$ and its low point, $l(v)$. Due to the simplicity of the network in *Figure 4*, it is clear that the network is a 2-node- (or a 2-edge) connected graph and therefore an s-t numbering is possible. In the case of a large poorly connected network, the DFS defines all the bridges or cut-nodes. Utilizing this information the network can be easily converted into a 2-node or 2-edge-connected graph.

Next, the s-t numbering algorithm is executed as outlined in Appendix 3. The algorithm provides a number to each node, $g(v)$, as shown in Figure 5. Notice that every node (not including t) is adjacent to at least two nodes, one with a higher number and the other with a lower number. Several schemes of s-t numbering can be found although the numbering relationships between the nodes are not altered.

Tree No. 1 is constructed according to the first phase of the 2-tree algorithm. For each node, v, where $v \neq t$, an edge (u, v) is chosen, such that $g(u) < g(v)$, and is directed from u to v. The candidate edges and the one selected are given in *Table 2*. The selection of the edges may be directed by the designer or according to the minimum distance tree. Assuming that all the edges are of equal length, Dijkstra's algorithm[13] may be used to define the minimum distance tree as shown in *Figure 5* by full lines.

Tree No. 2 is similarly delineated. For each node v, $v \neq s$ an edge (w, v) is chosen, such that $g(w) > g(v)$, and is directed from w to v. The candidate edges and the one selected are given in *Table 2*. Using the criterion of minimum distance tree, tree No. 2 is generated as shown in *Figure 5* by dashed lines. Examining the two trees it appears that each node can be reached by two node-disjoint paths and that every path belongs to a different tree. In case of failure of a single node or edge, at least one path between the source and every node will remain. The resulting network is therefore an invulnerable network under any single node or edge failure.

Network capacity adjustment

Once the topological invulnerability of the network is established, the next step is to ensure the hydraulic invulnerability of the network with a minimum cost. This is accomplished by adjusting the pipe diameters in the resulting network. The least-cost diameter set may be found by a straightforward application of the two trees approach. Recall that a necessary condition for the network to resist one pipe failure is that each tree will independently fulfil the hydraulic demands of the whole network. Since the flow pattern is uniquely defined for a tree network the constraint set consists of the head

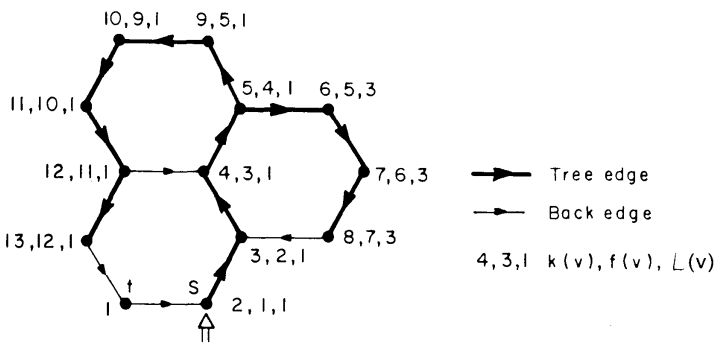

Fig 4 DFS on a schematic network, starting from node t

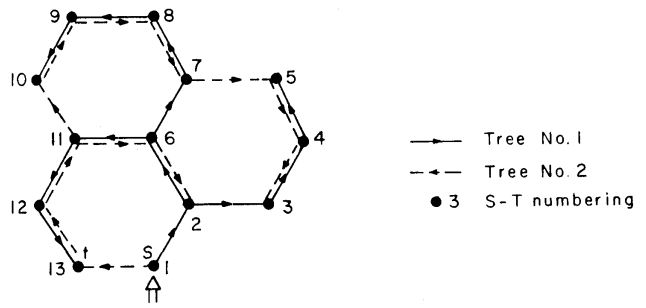

Fig 5 S-T numbering and the corresponding two trees (following the DFS in *Figure 4*)

constraints only, i.e. the minimum pressure in every node in each tree. The objective function is given by the sum of the pipe costs, usually as a function of the pipe diameter and the pipe length. The model described can be solved using linear programming. This approach was developed by Karmeli *et al.*[7] for branching networks, and then used by Alperovits and Shamir[2] for looped systems. This method makes use of a new variable, $x_{e,d}$, which is the length of the pipe segment in edge e with diameter d. The difference between the model used in this paper and the original model of Karmeli *et al.*[7] lies in the constraint set. While the original constraint set corresponds to a single tree the new constraint set includes the constraint sets of two different trees.

Consider a gravity fed network of a single source and $(nn-1)$ demand nodes for which the demands are specified. The two spanning trees are T1 and T2. There are $(nn-1)$ paths in each tree connecting the source to every demand node. Denote $P1(i)$ and $P2(i)$, $i=\{1, 2, \ldots, nn-1\}$, the set of paths in T1 and T2, respectively. Let $hmin_i$ be the minimum hydraulic head at node i and h_s be the hydraulic head at the source. The maximum allowed head drop in each path is given by $\Delta h_i = h_s - hmin_i$. The least-cost model is:

$$Cost = Min\left\{\sum_{e\in E}\sum_{d\in D_e} c_{e,d}x_{e,d}\right\} \tag{2}$$

subject to:

$$\sum_{e\in P1(i)}\sum_{d\in D_e} j^1_{e,d}x_{e,d} \le \Delta h_i \qquad i=\{1, 2, \ldots, nn-1\} \tag{3}$$

$$\sum_{e\in P2(i)}\sum_{d\in D_e} j^2_{e,d}x_{e,d} \le \Delta h_i \qquad i=\{1, 2, \ldots, nn-1\} \tag{4}$$

$$\sum_{d\in D_e} x_{e,d} = a_e \qquad \forall e\in E \tag{5}$$

$$x_{e,d} \ge 0 \tag{6}$$

where:

a_e length of edge e.
$c_{e,d}$ cost per unit length of a pipe segment with diameter d in edge e.
$j^1_{e,d}$ hydraulic gradient on the segment with diameter d in edge e of tree No. 1
$J^2_{e,d}$ hydraulic gradient on the segment with diameter d in edge e of tree No. 2.
$x_{e,d}$ length of a pipe segment with diameter d in edge e.
Δh_i head difference allowed between the source and node i.

and

$d\in D_e$ — set of possible commercial pipe diameters in edge e.
$e\in E$ — set of all edges in the network.

Equation (2) is the objective function, namely the minimum cost of the pipeline network. Equations (3) and (4) are the head constraints of tree No. 1 and tree No. 2, respectively. Equation (5) is required to ensure that the sum of the segment lengths in each edge equals the edge length. The hydraulic gradient $j_{e,d}$ is given by the Hazen-Williams equation:

$$j_{e,d} = \alpha\, chw_{e,d}^{-1.852}\, d^{-4.87}\, q_e^{1.852} \qquad \forall d\in D_e \tag{7}$$

where α is a numerical coefficient, whose value depends on the units used, and $chw_{e,d}$ is the Hazen-Williams coefficient of the pipe smoothness, q_e is the flow in edge e of a particular tree. An analogy exists between this model and the model suggested by Alperovits and Shamir[2]. In the last model the constraints are solved simultaneously for several demand patterns by combining their corresponding constraint sets into a single set of constraints.

An examination of the resulting network will reveal that the same node pressures will be above the minimum pressure as defined by the head constraints of the LP model. Since the first tree is capable of supplying the whole network, an addition of some supplementary edges (belonging to the second tree) will result in an increase in pressure at some of the nodes. The excess pressure can be reduced if under a single failure the node pressure are allowed to decrease to an acceptable level below the minimum level. Referring to the reduced level as an emergency water supply condition, the LP model can be adjusted to meet these conditions by reformulating the head constraints. Such a possibility reveals an attractive feature of the two-tree approach by which an economical analysis may be provided. That is, the LP model may be solved repeated for a number of emergency conditions so that a relationship between the network cost and the emergency conditions so that a relationship between the network cost and the emergency conditions (levels of service) may be defined. Given a particular budget limitation, the decision maker may then select the desired level of service.

Once the least-cost invulnerable network has been defined, it should be tested by a network solver to examine the network response to other demand patterns. Usually it can be expected that the additional patterns

Table 2 The tails of the candidate and the selected (*) edges to enter each node at *Figure 5*

Node number	1	2	3	4	5	6	7	8	9	10	11	12	13
Tree No. 1	—	1	2	3	4	2	6*	7	8	9	6*	11	12
	—	—	—	—	—	—	5	—	—	—	10	—	—
Tree No. 2	—	3	4	5	7	7	8	9	10	11	12	13	1
	—	6*	—	—	—	11*	—	—	—	—	—	—	—

will be satisfied with adequate pressures due to the network redundancy (as required by the reliability constraint). Still, this is no guarantee and the network should be tested accordingly. In case of insufficient pressure in a particular node it is recommended to increase the pipe size locally around the node vicinity and re-run the network solver.

The test of the network for additional demand patterns may be avoided if a common single source exists for all demand patterns, i.e. a network without operational reservoirs which become sources during the peak demand period. Having a common source, the same pair of trees is used to define the alternative paths in each demand pattern. Let $T1$ and $T2$ be the first and the second tree, respectively, and assume np demand patterns. There are np sets of path-head constraints which are defined by $T1$ (equation 3) and np sets of path-head constraints which are defined by $T2$ (equation 4). The total number of the path-head constraints is $2 \times np \times (nn-1)$. Solving the LP model for the enlarged set of constraints the optimal solution is the least-cost one satisfying the hydraulic conditions for all the demand patterns.

Summary of the methodology

Summarizing the proposed design procedure of an invulnerable least-cost water distribution network, the stages are:

1. Define the correct connectivity measurement of the network (2-edge or 2-node-connected graph). Assume the initial layout of the network.
2. Choose the critical pattern of flow demands based on maximum demands and reliability consideration.
3. Test the assumed layout for its connectivity degree using the DFS algorithm (Appendix 2). Add edges to the graph if necessary.
4. Label the nodes of the graph according to the s-t numbering (Appendix 3).
5. Identify tree No. 1 and tree No. 2 using the 2 trees algorithm. To choose between the various possible pairs of trees it is recommended that the Dijkstra algorithm be used.
6. Solve an LP model to assign the least-cost pipe diameters to every edge.
7. Examine the network under various demand patterns using a network solver. Make minor adjustments where required.

Conclusions

The methodology presented in this work was motivated as a result of difficulties encountered in solving the overall design problem of water distribution networks. The difficulties associated with this problem arise from the range of demand patterns and failure conditions that the network must satisfy, the stochastic nature of the load patterns and the tendency of the least-cost solution to oppose any network redundancy as required by the reliability considerations.

The proposed methodology casts the reliability issue in terms of system invulnerability and thus avoids the need to calculate the actual level of network reliability by attributing a failure probability for each hydraulic element and calculating the probability for a breakdown in the water supply. The stochastic behaviour of the load patterns is addressed by realizing that the invulnerability requirement is generally the more dominant factor in view of the network capacity. Therefore, satisfaction of the invulnerability requirement usually requires feasible hydraulic solutions for the various load patterns.

Following the above assumption the recast problem is still too difficult to be solved directly. This is in accordance with the difficulties encountered in the least-cost solution of large looped networks. As an alternative, a heuristic approach is suggested in which the solution is directed by the designer. Following this approach the designer is supplied with a number of reliable and fast subprograms which are used in an interactive manner. These subprograms consist of graph theory algorithms, linear programming and a general network solver.

Further research is needed in order to determine the relation between the pressure redundancy and the desired level of service under emergency conditions. Also it is recommended that the risk involving a 2-edge-connected network be compared with the risk involving a 2-node-connected network. This work assumed that the water supply is always available from a single source. Such an assumption is not satisfied in the case of a number of sources, each subjected to possible failure. An enlarged invulnerability approach will be required to include invulnerable supply at the various sources.

References

1 **Goulter, I. C.** Current and future use of systems analysis in water distribution network design, *Civ. Engng. Syst., 1987, 4 (4), 175–181.*

2 **Alperovits, A. and Shamir, U.** Design of optimal water distribution systems, *Water Resour. Res., 1977, 13 (6), 885–900*

3 **Morgan, D. R. and Goulter, I. C.** Least cost layout and design of looped water distribution systems, in *Intr. Symp. on Urban Hydrology Hydraulic and Sediment Control, University of Kentucky, Lexington, 27–29 July, 1982*

4 **Goutler, I. C. and Coals, A. V.** Quantitative approaches to reliability assessment in pipe network, *J. of Transportation Engng., ASCE, 1986,* **112** *(3), 287–301*

5 **Rowell, W. F. and Barnes, J. W.** Obtaining layout of water distribution network, *J. Hydraulics Div., ASCE, 1982,* **108** *(Hy 1), 137–150*

6 **Su, Y. C. Mays, L. W. Duan, N. and Lansey, K. E.** Reliability-based optimization model for water distribution systems, *J. of Hydraulic Engng., ASCE, 1987,* **113** *(12), 1539–1556*

7 **Karmeli, D. Gadish, Y. and Meyers, S.** Design of optimal water distribution networks, *J. Pipe. Div., ASCE, 1968,* **94** *(PL1), 1–10*

8 **Even, S.** 'Graph algorithms', *Computer Science Press, Potomac, Maryland, 1979, 62–63*

9 **Itai, A. and Rodeh, M.** The multi-tree approach to reliability in distributed networks, *Found. of computer science, Proc. of the 25'th symp., 1984, 137–147*

10 **Lempel, A. Even, S. and Cederbaum, I.** An algorithm for planarity testing of graphs, in *Theory of Graphs—Inter. Symp., Rome, July, 1966, Gordon and Breach, NY, 1967, 215–232*

11 **Even, S. and Tarjan, R. E.** Computing an s-t numbering, *The Computer Science, 1976,* **2,** *339–344*

12 **Bhave, P. R.** Noncomputer optimization of single source networks, *J. of Environ. Engng., ASCE, 1978,* **104** *(EE4), 799–814*

13 **Deo, E.** 'Graph theory with application to engineering and computer science', *Prentice-Hall, Inc., Englewood, NJ, 1974, 292–293*

Appendix 1: Graph theory terminology

An *undirected graph* $G(N, E)$ consists of a set of nodes, N, and a set of edges, E, where each edge corresponds to an unordered pair (u, v) of nodes. A *directed graph* $G(N, A)$ consists of a set of nodes, N, and a set of directed edges, A,

where each directed edge corresponds to an ordered pair of nodes. A *directed edge* (u, v) has *head*, u, and *tail*, v, such that the edge leaves u and enters v. Since many properties of an undirected and directed graph are the same, it is often convenient to disregard the orientations of the edges. The undirected graph which is obtained from a directed graph G is named the *undirected graph corresponding to G*.

$\hat{G}$ is a *subgraph* of G if the node and edge sets of $\hat{G}$ are subsets of the node and edge sets of G. $\hat{G}$ is a *spanning subgraph* of G if both their node sets are equal. A graph G is *complete* if, given its node set, its edge set contains all possible edges.

A *path* from v_1 to v_n, in an undirected graph, is a sequence of edges $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$. A *circuit* is a path such that $v_1 = v_n$ and a *simple circuit* is a circuit whose nodes are distinct. *Directed path* and *directed circuit* are analogously defined in a directed graph. A *weakly directed path* in a directed graph is a path in the corresponding undirected graph, but is not a directed path. Paths are *edge-disjoint* or *node-disjoint* if they consist of distinct edges or nodes, respectively. An undirected graph is *connected* if, for any two nodes u and v, there is a path from u to v. A *component* is the maximal connected subgraph in which a particular node is included. A *disconnected graph* consists of more than a single component.

A graph is *acyclic* if it contains no circuits. A *tree* is an undirected, connected, acyclic graph whose edge called *branches*. A *spanning tree* of a graph is a spanning subgraph which is a tree. An *arborescence* is an acyclic directed graph with one node, the *root*, having no entering edge, and all other nodes having exactly one entering edge. A *spanning arborescence* of a directed graph G is a spanning subgraph which is an arborescence (also called a *directed spanning tree*).

A *cut-set* is defined as a set of edges whose removal from a connected graph leaves the graph disconnected, provided that no other subset of these edges disconnects the graph. A cut-set can be identified between any two mutually exclusive subsets of nodes by means of the minimum number of edges whose removal destroys all paths between these two subsets. The number of cut-sets in a graph is usually much higher than the number of nodes, with an upper boundary equal to the number of different node partitions. One special cut-set of particular interest is the one by which a node is isolated from all others. The number of edges in such a cut-set is equal to the number of edges which meet that particular node. This number is known to be the *node degree*.

A *network* is a graph in which all the edges possess some properties in addition to their two end nodes. In case of water distribution networks edges represent hydraulic elements, which are mostly pipelines, characterized by their length diameter and roughness. Following water engineering terminology the term *loop* means a circuit while a *looped* network means a connected graph in which each edge belongs to a circuit (flow is permitted in both directions). Flow into and out of the network is assumed to be concentrated at nodes.

Appendix 2: The DFS algorithm

The DFS (Depth First Search) is widely recognized as a powerful technique for solving various graph problems. It consists of a scanning procedure over an undirected graph by which every node is numbered and every edge is directed. A directed spanning tree is identified, dividing the edges into *tree edges* and *non-tree edges* called *back edges*. The following version of DFS is after Even[8], and is used to detect bridges and cut-nodes.

Consider an undirected and connected graph $G(V, E)$ of (nn) nodes and (ne) edges. The algorithm assigns a number $k(v)$ for every $v \in V$ starting from an initial node s. The node $f(v)$, where $v \neq s$, is the least number, $k(u)$, of a node u which can be reached from v via a directed path, consisting of tree edges, followed by one back edge. The algorithm uses a *stack*, S, to restore the nodes found in each nonseparable component (2-node-connected or more). In the following algorithm the symbol, $\leftarrow$, represents an assignment operation.

1. Mark all the edges 'unused'. Empty the stack S. For every $v \in V$ let $k(v) \leftarrow 0$. $i \leftarrow 0$.
2. $v \leftarrow s$.
3. $i \leftarrow i+1$, $k(v) \leftarrow i$, $L(v) \leftarrow i$ and put v on S.
4. If v has no 'unused' incident edges go to Step (6).
5. Choose an 'unused' incident edge $e=(v, u)$. Mark e 'used'. If $k(u) \neq 0$, then:
   Let $L(v) \leftarrow \min \{L(v), k(u)\}$ and go to Step (4).
   Else
   Let $f(u) \leftarrow v$, $v \leftarrow u$ and go to Step (3).
   Endif
6. If $k(f(v)) = 1$ then:
   Go to Step (8)
   Else
   If $L(v) < k(f(v))$ then:
   $L(f(v)) \pm \min \{L(f(v)), L(v)\}$ and go to Step (7)
   Else
   $f(v)$ is a cut-node. All the nodes on S down to and including v are now removed from S. This set, with $f(v)$, forms a nonseparable component.
   Endif
   Endif
7. $v \leftarrow f(v)$ and go to Step (4).
8. All nodes on S down to and including v are now removed from S. They form with s a nonseparable component.
9. If s has no 'unused' incident edges then:
   halt.
   Else
   Node s is a cut-node. Let $v \leftarrow s$ and go to Step (5).
   Endif

The algorithm is of time complexity $O(ne)$ since each edge is scanned exactly once in each direction. As a result, the number of operations per edge is bounded by a constant.

Having the $f(v)$ numbers the directed spanning tree is defined by all the directed edges $(f(v), v)$. In case of a disconnected graph the algorithm halts after scanning the component which belongs to the starting node. More components may be discovered by reapplying DFS from new starting points not included in the previous components. Notice that while the cut-nodes are given directly by the DFS, the bridges are the edges which connect any two cut-nodes (assuming only one edge exists between two adjacent nodes).

Appendix 3: The s-t numbering algorithm

The algorithm described here finds the *s-t* numbering of a nonseparable graph following the works of Even and Tarjan[12] and Itai and Rodeh[10]. Let $G(V, E)$ be a nonseparable graph with a single source (s) which is adjacent to another node t such that (t, s) exists. Assume that DFS has been already executed with a starting node t and first edge $\langle t, s\rangle$ ($k(t)=1$ and $k(s)=2$). For each node $v \in V$ the DFS computes its number $k(v)$, its *father* $f(v)$, its *lowpoint* $L(v)$, and separates between the tree edges and the back edges.

The algorithm uses a *stack* to process nodes in the reverse order in which they were encountered. Initially, only (t) and (s) are on the stack with (s) on the top. Nodes which were never on the stack are considered 'new'. The algorithm finds the *s-t numbering*, $g(v)$, as follows:

1. Set $i \leftarrow 1$.
2. If the stack is empty then halt.
3. Remove the top node v from the stack.
4. $g(v) \leftarrow i$, $i \leftarrow (i+1)$.
5. For all tree edges $\langle v, w\rangle$ to a 'new' node w do:
   Trace the path whose first node is w and from there it goes up the tree and ends with one back edge into the node $L(v)$. Let the path be $\{(w=w_1), w_2, \ldots, w_m, L(v)\}$. Push $w_m, w_{m-1}, \ldots, w_1$ into the stack (w_m first).
6. For all back edges (u, v) to a 'new' node u do:
   Trace a path whose first node is u and from there it goes backwards via the tree edges until it encounters an *old* node, (o). Let the path be $\{(u=u_1), u_2, \ldots, u_m, o\}$. Push $u_m, u_{m-1}, \ldots, u_1$ into the stack (u_m first).
7. Go to Step (2).

The time complexity of the algorithm is $O(ne)$ where ne is the number of edges in the graph. An example of this algorithm is given in *Figure 4*.