AUTOMATIC MINIMAL LOOP EXTRACTION AND INITIALISATION FOR WATER PIPE NETWORK ANALYSIS

KAILASH JHA

Faculty-in-Charge CAD/CAM Center, Dept. Of Mechanical Engineering and Mining Machinery Engineering
Indian School of Mines University Dhanbad-826004 (India), Kailash_jha@hotmail.com

Abstract: In the present research new algorithms for automatic minimal loop extraction and initialisation have been developed for water pipe network analysis. The present work converts water pipe networks into graph data structures with pipes as edges and nodes of the network as nodes of the graph. The Nested Breadth First Search (NBFS) algorithm has been developed for the automatic extraction of loops from the graph. The First Breadth First Search (BFS) gives the clue, as back edges for loops are finally extracted by applying another BFS with the help of vertices of the back edge. Heuristics have been developed to get minimal loops. Pipes are initialised with the help of extracted loops. Water pipe networks are analysed using the Hardy Cross equation for balanced discharges. The present technique can be used for leak detection and pressure determination in a pipe network in real-time.

Keywords: NBFS, Hardy Cross equation, Loop detection, Water pipe network analysis, Darcy-Weisbach equation.

1. INTRODUCTION

In the present research water pipe network analysis has been done for a given network. This work has been carried out in three steps. In the first step the given network is converted into a graph data structure and the loops are extracted by the graph traversal (to travel a graph is to process every node in graph exactly once). Pipes are initialised in the second step using extracted loops. The network is finally analysed in the third step using the Hardy Cross technique (Cross, 1936). The present loop extraction algorithm is based on the graph traversal. There are two basic graph traversals: (1) Depth First Search (DFS), and (2) Breadth First traversal (BFS). Depth first search traversal follows a single path to its end, then the back tracing starts. In DFS, edges are explored out of the most recent discovered vertex (v) that still has unexplored edges leaving it. BFS traversal visits all the children before the first grand children. BFS search expands the frontier between discovered and undiscovered vertices across the breadth of the frontier. A graph network has been shown in figure 1, which has eight vertices. The order of vertices in DFS and BFS traversals for the network shown in Figure 1 are 1, 2, 5, 6, 3, 7, 4, 8 and 1, 2, 3, 4, 5, 6, 7, 8 respectively. In the present work, the Nested Breadth First Search (NBFS) technique has been developed to determine minimal loops in the given network. In the first BFS, signature edge (back edge) is identified and loops are extracted by second BFS traversal. Loops obtained in this work are the minimal loops, which have no overlapping and hence cannot be written as a sum of the shorter loops. The given network is assumed in this work as a graph data in which pipes are considered as the graph edges and nodes are considered as graph nodes. BFS is a simple algorithm for graph traversal. The graph considered in present formulation is a connected undirected graph and it is represented by adjacency list (a closely related data structure for representing graph). It explores systematically the edges of the graph to discover every vertex reachable from the start node.

The pipes extracted in loops have been initialised interactively to get the initial discharges for Hardy Cross technique (Cross, 1936) for water pipe network analysis. During initialisation, the continuity equation is applied at all nodes. The continuity equation says that the summation of all the values of discharges at a node must be zero. If at any node, this equation is not satisfied then the network is unbalanced. Pipe initialisation is major problem in loop based network analysis. The present work gives an interactive initialisation because in some cases changes in the order of the extracted loops are needed. If at least one node of a loop does not find the initialised adjacent pipe it cannot be initialised without reordering the extracted loops. In the present initialisation, supply pipe is initialised at last to adjust the final unbalanced discharge. An example of a network including two loops is shown in figure 2. Loop number 1 comprises of nodes 4, 3, 7 and 6 and loop number 2 comprises of nodes 1, 5, 4, 6, 7 and 2. If the order of loop initialisation is 2 and 1 respectively then loop number two can be initialised easily but loop number one cannot be initialised without changing the order of the initialisation because it does not have an uninitialised adjacent edge. In this case the order of loops is performed to get an initial solution for the Hardy Cross technique (Cross, 1936) for water pipe analysis. Loops 1 and 2 are, therefore dependent and

independent respectively for the network shown in figure 2.

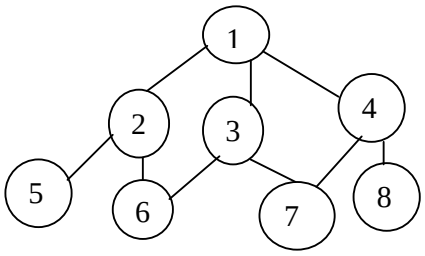

Figure 1: Illustration for graph traversal

The Hardy Cross (Cross, 1936) technique has been used for water pipe analysis for a long time. It is a simpler technique, which has been initially used to get

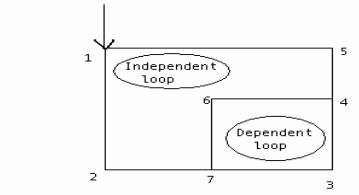

Figure 2: Dependent and independent loops

manual solution. In the present work, a water network has been analysed and the balanced discharge in pipes has been obtained. In automatic minimal loop detection, first the signature edge for the loop, which is the back edge, is determined. The basic clue behind the signature edge is that its vertex is repeated in the course of BFS traversal. Once the signature is obtained, another BFS is called to get a path from one signature vertex to another signature vertex. The path travelled by the second BFS gives the cycle/loop. This process is repeated for all the signature edges. If any signature edge is not able to produce the cycle, then the whole program is repeated with a different starting node. If all the signature edges of a starting vertex give a loop then this algorithm successfully terminates. All the loops are stored in a data file. If no starting vertex gives all the possible loops, the algorithm terminates with the message that division of the graph is needed before restarting the current algorithm. The BFS traversal gives the shortest path of edges in the traversal. This property of BFS helps in getting the minimal cycles. Stack implementations have been done for both the traversals. Heuristics have been used to get the required cycles. Heuristics are based on the concept that a bounding edge cannot be part of more than one cycle. A non-bounding edge may be common to more than one cycle. A cycle will be accepted only if all its edges have been visited by first traversal. Loops are reordered if the initialisation is not successful. The loop, that contains the supply node, is initialised last. If more than two supply nodes are present in the network, any loop containing the supply node is considered as the last loop and corresponding pseudo loops are added in the network. Reordering of the extracted loops should be such that all loops can be initialised. The initialisation algorithm includes the condition that pipe initialisation should end on the vertex of the loop containing an uninitialised adjacent edge. This algorithm can be used not only for pipe networks but also for electrical networks, structural flexibility analysis and in chemical structure storage and retrieval systems. Minor losses in pipes have not been considered in the present pipe network analysis. Leak detection and pressure determination in the pipe network can be done by present formulation in real-time with few calibrated nodes. Optimal pipe selection can be done on the basis of the pressures and discharges in the network. The present work can be used as a design tool as well as a real time monitoring tool for water pipe networks.

The remainder of the paper is organized as follows. Previous work is presented in section 2. Current work is described in section 3. Results and discussions are included in section 4. Conclusions are presented in section 5.

2. PREVIOUS WORK

A survey of the literature in the areas of loop extraction as well as pipe networks is proposed in this section. Cerna and Pelanek (Cerna and Pelanek, 2002) have developed the distributed scalable algorithm for explicit fair cycle detection. In this technique, fair cycle is first detected and then the algorithm produces a cycle, which is in general shorter cycle than the Nested Depth First Search technique (NDFS). NDFS-based techniques take shorter time because they are able to detect cycles on-the–fly. Many processors cannot be used in Nested DFS technique because it cannot exploit parallelism due to its sequential nature. Cerna and Pelanek (Cerna and Pelanek, 2002) have used a set based approach, which works in a way similar to BFS and hence exploits parallelism. Barnat et al. (Barnat et al., 2002) have developed an algorithm based on distributed NDFS, which decomposes the graph into the sub graphs. Graph is first decomposed into the strongly connected components and then the partition is done according to the decomposition. Decomposition of the graph solves the verification problems. NDFS is applied on decomposed graphs to search only those paths that really form cycles in the graph. A static partition function is used to decompose the graph. Cycles are accepted by NDFS algorithm. The present work also needs decomposition of graph to get correct cycles from network. Brim et al. (Brim et al., 2001) developed NDFS based technique, which located graph with negative length in directed graph. Clarke et al. (Clarke et al., 1999) have used partial order reduction, which involves verification of representation exceptions only instead of whole

system behaviour. These representatives are chosen automatically during the traversal of the graph by exploring only a subset of all possible successors of a state. Moreover, to ensure the correctness of the reduction at least one state on every cycle has to be fully expanded.

Courcoubetis et al. (Courcoubetis et al., 1992) have developed the NDFS to compute acceptable cycles in which the first search is used to find the reachable acceptable state, while the second search is used to initiate the discovery of accepting state and detect the reachability of an acceptable state. This is a sequential technique, which is not suitable for parallel and distributed processing. Lafuente (Lafuente, 2002) has used a simplified distributed technique to get acceptable cycles in a state transition graph. A partition function is used to localize the cycle. NDFS has been used in this technique. Achieving scalability and good load distribution is the main problem with this work. Horton (8) has developed the first polynomial algorithm to find the minimal cycle basis and analyse the complexity of the algorithm. Vismara (Vismara, 1997) extended the work of Horton (Horton, 1987). The set of cycles corresponding to the minimal basis is not relevant for analysing the cyclic structure of graph. This restriction is due to the fact that the minimum cycle basis is generally not unique. The smallest canonical set of cycles, which describes the cyclic structure of graph, is the union of all the minimal cycle bases. Barnat et al. (Barnat et al., 2003) have developed a practical parallel on-the-fly algorithm for enumerative Linear Temporal Logic model checking. The computational load is distributed among several workstations, which communicate via a message passing interface.

Szule et al. (Szule et al., 2005) have used the Virtual Distortion technique to locate the leak in a network. Gradient-based optimisation has been used in this work. It identifies the location of the leak even in multiple locations. Water distribution network analysis using Excel has been done by David et al. (2004). In this work the Hardy Cross equation has been used. Arsene et al. (2004) have done modelling and simulation of water based system based on the loop equations. A co-tree simulation algorithm has been developed by Arsene et al. (2004) based on the decomposition of the water network in spanning trees and leakages, which are model in this work as addition demand between end nodes of pipe. Abebe and Solomatine (1998) have developed a method for the global optimisation of pipe networks in which the cost of a network is minimized for both the static and dynamic loading.

In Barnat et al. (2003) a similar loop detection is done in which signature edges are back level edges, obtained by parallel level-synchronized BFS and parallel checking is done by DFS to ensure at least one back-level edge for a cycle. Part of the present work has been described in (Jha, 2006).

.

3. CURRENT WORK

3.1 Automatic Minimal Loop Extraction

In the present work a new Nested Breadth First Search (NBFS) algorithm has been developed to obtain the minimal loops in a given network. The first BFS (BFSCycleDetection (int node)), determines the signature edge of loop and the second BFS (BFSCycleExtraction (int node1, int node2)) extracts loop with help of the signature edge. The signature edge is a back edge, which has two signature vertices. Once a back edge is encountered in the first BFS traversal, the second BFS traversal is invoked to extract the loop. The second BFS starts with the first signature vertex and the second vertex is searched. If the search is successful, it gives a list of vertices, which form the loop. Traversal through the signature edge is restricted. A link between the signature vertices is established to complete the cycle. The number of times an edge participates in loop extraction is determined by a tag (Cycled-Edge-Tag) value. The Cycle-Edge-Tag is initialised at zero. It is incremented once a cycle has been extracted. If the Cycled-Edge-Tag value for a boundary edge is greater than one, it will not take part in further cycle extractions, because a boundary edge cannot form more than one loop. The signature edge depends on the starting vertex. If the edge is not marked as travelled by first BFS traversal it will not take part in loop extraction. The order of the loop extraction depends on the starting vertex. If the starting vertex gives all the loops, the program terminates after storing all loop information. In the current work, all the vertices of the graph are recalled in a for-loop for successful start vertex. If the start vertex gives all the possible loops other vertices are not attempted. If no vertex gives all the possible loops then the program terminates with the message that the current graph network needs to be divided before firing the same algorithm again.

```
Algorithm of minimal loop extraction
Int LoopExtraction ()
{
For each node s of network
{
            If (BFSCycleDetection(s) == 0)
              Continue
           Else
            Return 1
}
Message (given network need to be divided before
running this program)
Return 0
}

Int BFSCycleDetection (int node)
{
Put node s in stack
```

```
Put node s in stack-Travelled
While (stack is not empty)
    If (edge is tagged) Continue
    Else tag the edge
    Get the vertex
If (vertex is already in stack-Travelled)
        {
     If (BFSCycleExtraction (node, vertex) ==0)
                        Return 0
          }
       Put the vertex in stack
       Put vertex in stack-Travelled
  }
   Remove front node from stack
}
Return 1
}

int BFSCycleExtraction(int node1, int node2)
{
     Put node1 in stack1
      Int kkk =0
    While(stack1 is not empty)
      {
         Get the front node from stack1
         Get adjacent edges from front node
       For( each edge)
          {
            If (edge tagged by second traversal) Continue
            Else tag the edge
            If (edge is not travelled by first BFS traversal) Continue
             If (Cycled-Edge-Tag value ==1 && edge is boundary) Continue
                  Get next-Vertex
                  If (next-Vertex == node2)
                      {
                        MakeCycles()
                        Cycled-Edge-Tag++
                        for all cycled edges
                          kkk++
                        Break
                       }
              Put next-Vertex in stack1
            }
       Remove front node from stack1
       }
If( kkk ==0)            // Program is not going inside makeCycle()
       Return 0
Return 1
}
```

3.2 Interactive Initialisation

In the interactive initialisation, the pipes in the loops have been initialised to an arbitrary value of discharge satisfying continuity equation. In initialisation, the continuity equation has been checked at all the nodes of the loop. The continuity equation means that the sum of all discharges at a node should be equel to zero. The number of initialised and uninitialised pipes are determined at each node, and uninitialised pipes at the node are initialised by the continuity equation at the node. A fundamental condition for loop initialisation is that it should end at a node in the forward traversal where it must have an uninitialised adjacent edge. An adjacent pipe of start-node of the first loop is initialised with an arbitrary value if no discharge at this node is known. If a loop node has an initialised edge, start with this node and go on initialising in forward direction, before coming across the node having an unititialised adjacent edge. This node (stop-node) is initialised at last. The next initialisation of nodes in the loop is performed in the backward direction of the loop until the end-node is reached for initialisation. This process is carried out for all the loops. The supply vertex is initialised last to compensate the total unbalanced discharge. The starting node of the automatic loop extraction should be such that the first loop must contain the supply node. This condition is achieved by extracting first the loop containing the supply node and keeping it at the end of the loop list. If this does not work then loops need to be reordered. A loop should be initialised only when it contains an adjacent uninitialised pipe. If this condition is not satisfied then the loops need to be reordered. If the loops are not properly ordered then the initialisation cannot be performed. If the initialisation of the dependent loops shown in figure 2, starts with node 3, edges 3-7 and 4-3 are initialised first. Initialisation stops when it reaches node 7 in the forward direction because this node has an uninitialised adjacent edge. The initialisation of the nodes numbers 4 and 6 are performed in the backward direction. Node 7 is initialised last to pass unbalanced discharge through unitilialized edge 7-2. Nodes in the independent loops of figure 2 are initialised similarly. Node number 1 is initialised last to compensate total unbalances. If the initialisation of the independent loop is done first then the dependent loop cannot be initialised correctly. In figure 2, if the initialisation of the independent loop (loop 1) is done before that of node 6 of the dependent loop (loop 2) this will remain unbalanced due to the unavailability of uninitialised adjacent pipe. This algorithm gives an initial solution for further analysis of the network (Cross, 1936; David and Huddleston, 2004).

```
Algorithm of loop initialisation
InitialiseLoops()
{
GetLoops()
KKKK:
   Get first node (start-node) from the first loop
   Get all the adjacent edges
   Initialised an adjacent edge by an arbitrary value
   Tag the edge for initialisation
   For each loop
    {
          Find the node (start-node-for-loop) having initialised adjacent edge
               If (there is no such node)
```

```
            {
                Reorder the loops
                Go to KKKK
            }
```

Starting with this start-node-for-loop go on initialising the nodes in loop in the forward direction using the continuity equation until a node (stop-node) is found with an uninitialised adjacent edge

```
If (there is no such a node)
            {
    Change the order of extracted loops
    Go to KKKK
            }
        Go on initialising nodes in the backward direction until the stop-node is reached
    }
Return 1
}
```

3.3 Water Pipe Network Analysis

The algorithms described in previous sections give the extracted loops and the initial discharges for water pipe network analysis by the Hardy Cross method (Cross, 1936). There are two basic techniques to perform the water pipe network analysis: (1) the Nodal technique and (2) the Loop based technique. The Nodal technique needs the solution of equations. If large number of equations and unknowns are present (usually is the case with large networks), instability may arise due to round-off error. The problem becomes critical when these two numbers are not the same. The Loop based technique drastically reduces the number of unknowns. In the early days, due to its simplicity, this technique has been used as manual technique for the analysis of water pipe networks. In the current work, balanced discharge in the pipes have been obtained by using the Hardy Cross (Cross, 1936) technique. The basic theory behind loop-based techniques (Cross, 1936) is that the total head loss in a loop is zero. This is an iterative technique in which the incremental discharge is calculated with the help of discharges obtained by the previous iteration.

The total head loss in pipe according to the Darcy-Weisbach equation is

$$h_L = K * Q^n \qquad (1)$$

Where $K = 8*f*L / (\Pi * \Pi * g * D^5) + K_m$

L = length of pipe.
D = diameter of pipe.
f = coefficient of friction(0.033)
g = acceleration due to gravity.
n = 1.85(smooth pipe)
K_m is a constant introduced to account for minor losses

In the current application minor losses are ignored.

According to Hardy Cross the total head loss in a loop is

$$\sum h_L(Q) = \sum K Q^n = 0 \qquad (2)$$

In any pipe $Q = Q_0 + \nabla Q$

Q = final discharge in pipe
Q_0 = initial discharge
∇Q = incremental value of discharge

Substituting the expression for Q in equation 2

$$\sum h_L(Q) = \sum K * (Q_0 + \nabla Q)^n = 0;$$

The binomial expansion of the above equation yields

$$\sum h_L(Q) = \sum K * (Q_0^n + n Q_0^{n-1} \nabla Q + \text{-----}) = 0 \qquad (3)$$

$$\nabla Q = - \sum K * (Q_0^n) / (\sum K * n * (Q_0^{n-1})$$

$$\nabla Q = - \sum h_L / \sum (h_L / Q_0) \qquad (4)$$

$$Q_{modified} = Q_0 + \nabla Q \qquad (5)$$

Iteration is continued until incremental discharge (∇Q) is more than given value (0.0000001 m^3/s) for any loop, which may be lowered further if needed.

4. RESULTS AND DISCUSSIONS

Algorithms of automatic minimal loop extraction and initialisation have been developed for water pipe network analysis. Water pipe networks have been analysed by using the Hardy Cross technique. A number of cases have been tested. Results have been displayed by using OpenGL on a PC using Window operating system. C++ has been used for the development of the current work. Figure 3 shows an example (test result 1) of a network, wherein, nine loops have been successfully extracted. Nodes have been represented as dots and pipes as edges. Figure 4 shows the loops extracted from the network shown of figure 3. Loop directions have been shown for many pipes by single arrows because the directions of loops including these pipes are same(see figure 4). The supply pipe is pipe number 29, which is shown in figure 5. The automatic extraction of loops starts with the supply node or with a node near the supply pipe because the loop extracted first must contain the supply node. The first extracted loop is kept as the last loop to be initialised to compensate the total unbalances in the network. The loop extracted next is kept for initialisation before the first extracted loop. In the same way other loops are extracted and kept for the initialisation. Since initialisation is done in the reversed order of the loop extraction, hence the- loop extracted last is initialised first. Figure 5 shows the initial direction of the flows in a proportion that reflects the thickness of the pipes: these are obtained through the current initialisation algorithm. Pipe numbers are also shown in figure 5. Table 1 in Appendix A gives the details of the different pipes for the network shown in figure 3 along with the magnitude of the initial and balanced discharges. Balanced discharges in pipes for this example have been shown in proportion to the thickness of pipe (see figure 6)

according to the Hardy Cross technique. Figure 7 shows another example (test result 2) of a water pipe network. Figure 8 shows the extracted loops for test result 2(TR 2), which has ten loops. The extracted loops have been numbered inside the loop as shown in figure 7. The arrows shown in figure 7 indicate the directions of flow in the extracted loops. If initialisation of the discharges in pipes is done without reordering the extracted loops, a message is displayed indicating that initialisation cannot be done without reordering the extracted loops. For example, if the loops are initialised in the reverse order of loop extraction, then loop number eight will be initialised after loops number ten and nine, respectively. In this situation the balance of the total discharges at node number 21 is not achieved because loop 8, which includes node number 21, does not have any adjacent uninitialised pipes to through the unbalanced discharge. If loop number 10 is interchanged with loop number 8 then the initialisation of the network can be successfully completed as shown in figure 9. The directions and thicknesses of the pipes in figure 9 show the directions of the flow and the proportional initial discharges, respectively. The arrows in figure 10 show the balanced discharges in the pipes corresponding to test result 2, which has been obtained using the current technique. The thickness of the pipes shown in figure 10 is proportional to the balanced discharge in pipes. The details of pipes and discharges for test result 2 are shown in table 2 (Appendix A). Table 3 in Appendix A shows the extracted loops for test results 1 and 2, respectively. Table 4 in Appendix A shows details of the nodes for test results 1 and 2. Because loop extraction is based on the BFS traversal, there may be situations in which a particular starting node will not give all loops correctly.

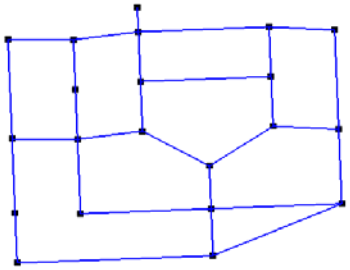

Figure 3: Water pipe network test result 1(TR 1)

In such cases, the algorithm can get the next starting vertex automatically for the extracted loops. If, after attempting all the nodes, the present algorithm fails to give the correct results then the given network should be decomposed according to the methods proposed in the relevant references (Barnat et al. 2002; Brim et al., 2001). Decomposed graph will give correct results. The implementation results are not shown in references (Brim et al., 2001; Clark et al., 1999; Courcoubetis et al. 1992; Lafuente, 2002) for cycle detection, only theoretical discussions are presented. In the current work, both cycle identification and extraction have been done by BFS traversal, which is a unique feature of this research. Partition of the graph and optimisation of the network are considered as future research developments. The Gradient-based technique

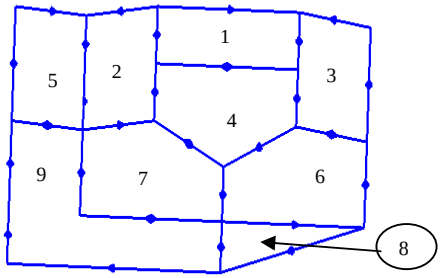

Figure 4: Extracted minimal loops with ID for TR 1

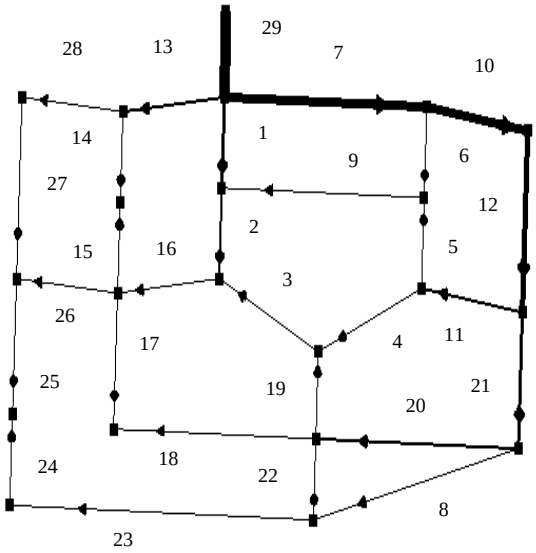

Figure 5: Initial proportional discharges with pipe ID

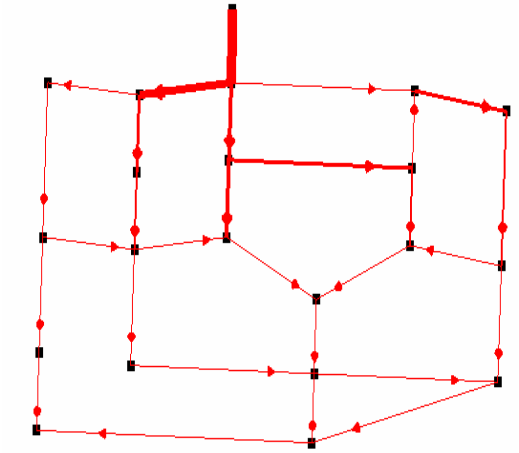

Figure 6: Balanced proportional discharges

can also be used to improve the result, provided that initial heads and initial discharges are known for the network. The present work can be used for leak detection and pressure calculation in the pipe network, which help in optimal selection of pipes for the network. The current work is very useful for the real-time analysis of pipe networks in terms of few calibrated nodes and can be used by the Municipal Corporation for water pipe network analysis and monitoring.

The present work can be used as a design tool as well as monitoring tool for water pipe network. The pseudo loops which comprised of the pseudo elements, and pumps in the network can easily be added in the present formulation (Streeter et al., 1997). The scales of the figures are not same due to the insertion of node IDs and pipe IDs.

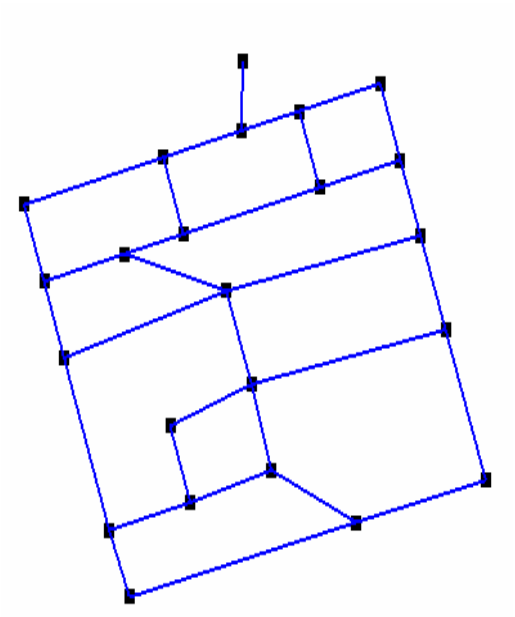

Figure 7: Network for test result 2 (TR 2)

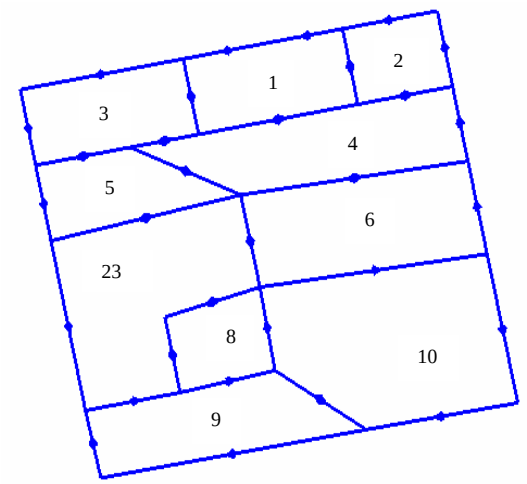

Figure 8: Extracted minimal loops with loop Nos. TR 2

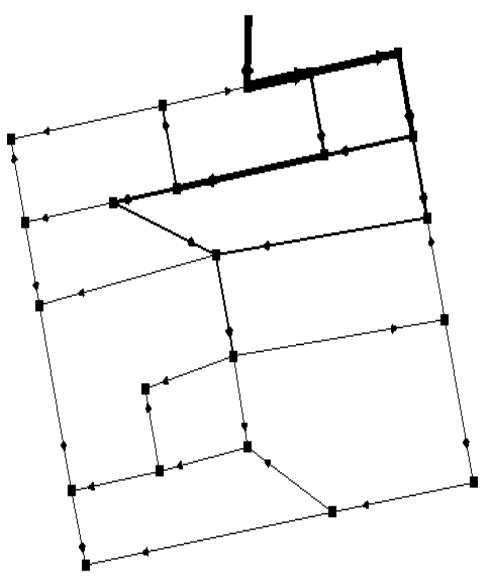

Figure 9: Initialised discharges in pipes for TR 2

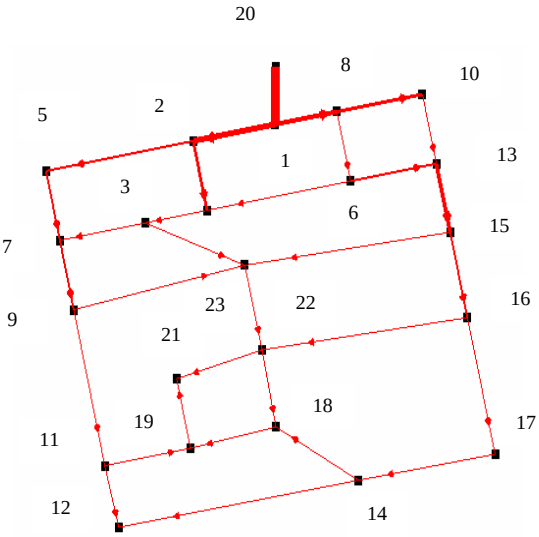

Figure 10: Balanced proportional discharges with node (ID)

5. CONCLUSION

Algorithms of automatic minimal loop extraction and interactive initialisation have been developed in this work for water pipe network analysis. Several results have been tested. Automatic minimal loop extraction is based on the Nested Breadth First Search algorithm. Heuristics have been used to get the correct loops. Loops extracted in this work are minimal loops. Cases have been identified where the ordering of loops is needed for the proper initialisation of discharge in pipes. The Hardy Cross technique has been used to get balanced discharge in water pipe networks. Minor losses in pipes have not been considered in the present formulation. No examples have been encountered where the division of the graph is needed for loop extraction. The present work can be used by the Municipal Corporation for designing as well as monitoring the water pipe network.

REFERENCES

Abebe, A. J. and Solomatine, D. P. (1998). ``Application of global optimisation to the design of pipe network", Proc. 3rd International conference on Hydro informatics, Copenhagen, August 1998, Balkema, Rotterdam.

Arsene C. T. C., Bargiela, A. and Al-dabass, D (2004). ``Modelling and Simulation of Water Systems Based on Loop Equations", I. J. of Simulation, Vol. 5, No. 1-2

Barnat, J., Brim, L., and Cerna, I. (2002). ``Property Distribution of Nested DFS", In proceedings of the 3rd International workshop on verification and computation logic (VCL'02 – held at the PLI 2002 Symposium), pages 1 –10. University of Southampton, UK, Technical Report DSSE TR-2002-5 in DSSE.

Barnat, J., Brim, Land Chaloupka, J. (2003). `` Parallel Breadth-First Search LTL Model-Checking", Proceeding of 18th conference on Automated Software Engineering (ASE-2003), Montreal, Canada, October.

Brim, L., Cerna, I., Krcal, P. and Pelanek, R. (2001). ``Distributed LTL model checking based on negative cycle detection", In R. Hariharan M. Mukund. And V. Vinay editors, Proceeding of foundation of software Technology and Theoretical Computer Science (FSt-TCS'01) Volume 2245 of LNCS, pages 49-73, Springer-Verlag.

Cerna, I and Pelanek, R. (2002). `` Distributed explicit fair cycle detection", In T. Ball and S K Rajamani, editors, Model checking software, 10 th International Spin workshop, Volume 2648 of LNCS, Pages 49-73, Springer-Verlag

Clarke, E M, Grumber, O. and Peled, D A (1999) ``Model checking" The MIT Press, Cambridge, Massachusetts

Courcoubetis, C., Vardi, M, Wolper, P. and Yannakakis, Y. (1992). ``Memory Efficient Algorithms for verification of Temporal properties", Formal Method in System design, 1:275-288.

Cross, H (1936). ``Analysis of flow in network of conduits of conductors", Bulletin No. 286, University of Illinois Engg. Experiment station, III.

David H. Huddleston, Vladimir J. Alarcon, and Wei Chen (2004). `` Water Distribution Network Analysis using Excel", J. of Hydraulic Engineering, Vol. 130, No. 10, PP 1033-35.

Horton, J. D. (1987). ``A polynomial-time algorithm to find the shortest cycle basis of the graph", SIAM J. Compu., 16(2): 358-366.

Jha, Kailash (2006). ``Automatic Loop Extraction and Initialisation for Water Network Analysis", Proceeding of 15th congress of APD-IAHR" IIT, Madras, India, August 7-10, 2006, pp. 1315-1323.

Lafuente, Alberto Lluch (2002). `` Simplified Distributed LTL Model Checking by localizing Cycles", Report no. 176, Institut Fur Informatik, Albert-LludWings, Freiburg, Germany.

Streeter, V. L, Wylie, E. B. and Bedford, K. W. (1997) ``Fluid Mechanics", McGraw-Hill Companies, 9th Edition.

Szulc, J., Kokakowski, P., and Nasher Nagib (2005). ``Leakage detection in water network", J. of

intelligent material systems and structures, Vol. 16, No. 3, PP 207-217.

Vismara, P. (1997). ``Union of all the minimum cycle basis of outer planar graphs", Electronic J. of Combinatory, 4(1), R#9.

BIOGRAPHY:

Dr. Kailash Jha is currently Faculty-in-Charge of CAD/CAM Center, Dept. of ME&MME, Indian School of Mines University, Dhanabd, India. He is an Assistant Professor in Dept. of ME&MME. He did his schooling in the BDAH School, Nanour (Madhubani). He received his BE (1991) and ME (1993) degrees from REC (NIT), Durgapur (WB), and his PhD (1999) from the Indian Institute of Science, Bangalore. His research interests include soft computing, surface modelling, feature-based design, geometrical reasoning and feature recognition.

Appendix A

Table 1: Details of pipes, initial and balanced discharges in pipes for test result 1

Pipe Index	Node 1	Node 2	Length (meter)	Diameter (meter)	Initial Discharge (meter^3/second)	Balanced Discharge (meter^3/second)
1	1	2	250	0.15	11.078	15.104
2	3	2	250	0.1	9.375	4.7618
3	3	4	312.5	0.225	0.9375	1.413
4	4	5	312.5	0.225	2.8125	4.4044
5	6	5	250	0.1	2.4063	4.4593
6	6	7	250	0.15	0.89063	0.88304
7	7	1	500	0.15	26.359	10.896
8	15	16	559.37	0.155	2.5	1.0591
9	2	6	500	0.225	0.29688	8.3424
10	7	8	253.18	0.155	23.469	9.7791
11	5	9	253.18	0.155	7.2188	1.945
12	8	9	500	0.155	20.469	6.7791
13	1	10	253.18	0.155	7.5625	19
14	10	11	250	0.155	1.2813	9.1527
15	11	12	250	0.155	0.71875	7.1527
16	12	3	253.18	0.155	4.3125	2.6512
17	12	13	375	0.166	0.875	3.086
18	13	14	500	0.166	1.125	1.086
19	4	14	191.72	0.166	1.125	2.8174
20	14	15	500	0.155	6.75	0.22504
21	9	15	375	0.155	11.25	2.834
22	14	16	250.8	0.155	2.5	1.6784
23	16	17	750	0.155	3	0.73747
24	17	18	250	0.155	1	1.2625
25	18	19	375	0.155	1	3.2625
26	19	12	250.8	0.155	0.71875	0.58448
27	19	20	500	0.155	2.2813	5.847
28	20	10	250.8	0.155	4.2813	7.847
29	1	21	180	0.45	45	45

Table 2: Details of pipes and initial and final discharges in pipes for test result 2

Pipe ID	Node 1	Node 2	Length (meter)	Diameter (meter)	Initial Discharges (meter^3/second)	Balanced Discharges (meter^3/second)
1	1	2	300	0.155	15.94	63.812
2	2	5	537.5	0.155	3.3001	24.877
3	5	7	250	0.155	0.11786	21.459
4	7	9	250	0.155	9.1916	21.565
5	9	11	562.5	0.175	11.815	12.624
6	11	12	253.16	0.185	2.2555	1.988
7	12	14	875.06	0.185	1.0715	1.339
8	14	18	353.69	0.185	1	0.51822
10	18	19	313.14	0.175	1.4267	1.373
11	19	11	312.5	0.175	0.31538	0.76099

12	19	21	250	0.155	1.1113	2.134
13	18	22	253.2	0.175	5.4387	5.8667
14	21	22	318.2	0.155	5.4387	4.416
15	22	23	312.66	0.155	32.632	11.012
16	23	9	626.98	0.155	9.1916	2.3727
17	23	15	751.42	0.165	26.687	1.8454
18	23	3	403.21	0.155	26.687	18.344
21	7	3	312.5	0.155	16.144	6.9404
22	3	4	225	0.155	48.433	30.887
23	4	6	525	0.155	83.295	7.5736
24	6	8	250	0.155	45.102	37.026
25	8	1	225	0.155	149.03	69.278
26	2	4	250	0.155	27.765	30.41
27	8	10	312.5	0.155	100.51	28.834
28	6	13	312.5	0.165	45.102	22.544
29	14	17	500.1	0.2	3.2145	3.0002
30	13	10	250	0.155	99.21	27.534
31	13	15	250.03	0.175	50.234	46.203
32	15	16	312.56	0.185	4.8732	15.938
33	16	17	500.1	0.2	9.7465	9.5322
34	16	22	751.67	0.2	14.62	6.4055
35	20	1	298.3	0.255	138.3	138.3

Table 3: Nodes of extracted loops for test results 1 & 2

	Node in the loops	
Loop Numbers in reverse order of extraction	Test result 1	Test result 2
1	17 18 19 12 13 14 16 17	14 18 22 16 17 14
2	16 14 15 16	14 12 11 19 18 14
3	13 12 3 4 14 13	18 22 21 19 18
4	15 9 5 4 14 15	21 22 23 9 11 19 21
5	19 20 10 11 12 19	16 15 23 22 16
6	4 3 2 6 5 4	15 13 6 4 3 23 15
7	9 8 7 6 5 9	23 3 7 9 23
8	12 3 2 1 10 11 12	3 4 2 5 7 3
9	6 2 1 7 6	13 10 8 6 13
10		6 8 1 2 4 6

Table 4: Details of nodes for test results

	Test result 1				Test result 2			
Node (ID)	Elevation	Demand	X-cord	Y-cord	Elevation	Demand	X-cord	Y-cord
1	20	0	-250	375	213.4	5.207	275	750
2	20	2	-250	125	212.9	8.525	-25	750
3	20	6	-250	-125	212.7	5.602	-250	500
4	20	3	0	-312.5	212.4	7.097	-25	500
5	20	2	250	-125	213.9	3.418	-562.5	750
6	20	3	250	125	212.2	6.908	500	500
7	20	2	250	375	212.7	6.835	-562.5	500
8	-20	3	500	375	212.45	3.418	500	750
9	-20	2	500	-125	212.1	6.568	-562.5	250
10	-20	2	-500	375	212.45	1.3	812.5	750
11	-20	2	-500	125	211.6	9.875	-562.5	-312.5
12	-20	2	-500	-125	251.5	3.327	-562.5	-562.5

13	-20	2	-500	-500	212	3.875	812.5	500
14	-20	2	0	-500	241.65	1.143	312.5	-562.5
15	-20	2	500	-500	215.75	28.42	812.5	250
16	0	2	0	-750	221.7	0	812.5	-62.5
17	0	2	-750	-750	231.7	6.532	812.5	-562.5
18	0	2	-750	-500	231.7	5.012	62.5	-312.5
19	0	2	-750	-125	211.65	0	-250	-312.5
20	0	2	-750	375	511.7	6.532	275	750
21	200	0	-250	375	211.9	6.55	-250	-62.5
22					271.85	7.135	62.5	-62.5
23					261.95	11.55	62.5	250