Low-Cost Technology for Water Utility Management

Sami J. Habib, Kuwait University, Kuwait
Paulvanna N. Marimuthu, Kuwait University, Kuwait

The Asian Conference on Sustainability, Energy & the Environment 2016
Official Conference Proceedings

Abstract
The amount of fresh water in the Middle East region is scarce due to long summer season with rising temperature, low precipitation, and lack of fresh water resources. Thus, fresh water supply depends mainly on non-conventional resources, such as desalination process, which converts the seawater into fresh water. The annual domestic-water consumption continues on an uptrend; and the rise is affected by diverse factors, such as the population growth, aging and poor maintenance of pipeline networks, and improper consumption. Further, the lack of up-to-date topology of existing water distribution network (WDN) and the present scenario of leakage detection by manual inspection necessitate the discovery of pipeline network through smart devices to minimize wastage and to monitor efficiently. In this paper, we have provided a soft-computing approach based custom-made WDN discovery tool, which aids in modeling the given WDN as a graph, and it explores the network graph utilizing classical breath-first and depth-first search algorithms.

Keywords: water distribution network; algorithm; discovery scheme; soft computing.

1. Introduction

The scarcity of natural water resources and the higher processing cost associated with seawater desalination impose the inevitability to manage the water distribution network (WDN) in Middle Eastern countries. WDN experiences many challenges from the frequent changes in its topology due to new installations and breakages. Furthermore, the unnoticed leakages and improper consumption of water urge the necessity for the development of soft-computing tools to monitor WDN. The soft-computing approach aims to develop a pipeline network discovery tool, where the network is represented as a graph, so that the topology of the network gets updated periodically.

The State of Kuwait depends on 6 major seawater distillation plants, which meet 95% of the fresh water demands. The statistical report from Ministry of Electricity and Water (MEW) documents the activities of WDN in Kuwait, which shows that the annual fresh water consumptions are increasing every year according to (MEW Report, 2007-2011). The statistics on number of fresh water consumers for a span of 8 recent consecutive years shows that the number of users is increased by 36.47%, which claims a periodical updation in WDN topology (Ghunaimi, 2005).

In this paper, we have proposed a custom-made interactive network discovery tool, which models the given WDN as a graph and it facilitates the discovery of location of an unknown node from a selected root node. We have utilized Java Universal Network Graph (JUNG) (Madadhain et al., 2005), a software library to model WDN as a graph and we have applied classical search algorithms, such as the breath-first search (BFS) and depth-first search (DFS) to traverse the graph.

2. Background

2.1 Network Discovery Tools

Water distribution pipeline monitoring research focuses on network discovery tools and hardware units to mitigate the water losses. Few researchers have designed network discovery tools (Prisco et. al, 2011), (Rashid et. al, 2013), (Baáut and Urbaniak, 2011) and simulation software (Yang, et al., 2015), (Paluszczyszyn, 2015) to monitor the water losses. Our intensive literature survey revealed that there were few commercially available design software packages for designing water pipeline network, such as HydrauliCAD (2013), STANET (1984), EPANET (Rossman, 2000), Pipe2000 (Frazey, 2004) and AquaNET (1992). The summary of the existing discovery tools are illustrated in Table 1:

Table 1. Summary of water network discovery tools.

Software Package	Limitations	Cost ($)
HydrauliCAD	Run as a part of AutoCAD and it does not include transient analysis	Free
STANET	Data can be imported from Geographical Information system (GIS) and it needs dedicated interfaces to handle the data	≈ $1900
EPANET	Suitable for pressurized pipe networks and the map may not be to scale	Free
AquaNET	Suitable for pressurized pipe networks and it works with GIS	≈ $2000
Pipe2000	Hydraulic modeling software for simple and complex pipe systems	≈ $1495

We have utilized Java Universal Network Graph (JUNG), a software library that provides a common and extendible language for modeling, analyzing, and visualizing of data that can be represented as a graph or network. It is written in Java, which allows JUNG-based applications to make use of the extensive built-in capabilities of the Java API, as well as those of other existing third-party Java libraries. Moreover, the JUNG architecture is designed to support a variety of representations of entities and their relations, such as directed and undirected graphs. It provides a mechanism for annotating graphs, entities, and relations with metadata. This facilitates the creation of analytic tools for complex data sets that can examine the relations between entities as well as the metadata attached to each entity and relation. Our tool utilizes two classical graph traversal algorithms, such as the breath-first search and depth-first search to traverse the generated WDN graph.

2.2 Graph Traversal Algorithms

A graph G (V, E) is a data structure, consists of a finite non-empty set of vertices (V) and a set of edges (E), which is used to model pairwise relations between objects in a network or in a system. Understanding complex systems often requires a bottom-up analysis, which can be done by examining the elementary constituents individually and then how these are connected. The myriad components of a system and their interactions are mainly represented as graphs, which may be explored using traversal algorithms. A traversal algorithm is a systematic walk, which visits the nodes in a specified order, which can be used for solving real-world problems such as discovering, searching, scoring, and ranking. We have considered the breath-first search and depth-first search to traverse the generated WDN graph.

2.2.1 Breath-First Search Algorithm

The breath-first search (BFS) discovers the graph layer by layer (level by level) to identify all of the connected vertices within a graph (Cormen, 1992). The BFS algorithm illustrated in Figure 1, assumes that the input graph *G (V, E)* is represented using adjacency lists; and BFS maintains several additional data structures with each vertex in the graph. The color of each vertex $u \in V$ is stored in a vector *color[u]*, and the predecessor of *u* is stored in a vector *π[u]*. If u has no predecessor (no parent such as the root), then *π[u] = NIL*. The distance from starting vertex *s* to vertex *u* is stored in *d[u]*. The algorithm uses first-in, first-out queue *Q* to manage the set of vertices.
In Figure 1, Lines 1-4 perform initialization such as coloring every vertex white, setting *d[u]* to be infinity for every vertex *u,* and setting the parent of every vertex to be *NIL*. In line 5, the starting vertex *s* is colored gray, since it is considered to be discovered initially. Line 6 initializes *d[s] to 0*, and line 7 sets the predecessor of the starting vertex to be *NIL.* Line 8 initializes the queue (*Q*) containing the vertex *s*; thereafter *Q* always contains the set of gray vertices.

```
Algorithm BFS(G,S):
1    Begin
2         for each vertex u ∈ V[G]- {s}
3              do color[u] ← WHITE
4                 d[u] ←∞
5                 π[u] ← NIL
6         color[s] ← GRAY
7         d[s] ←0
8         π[s] ← NIL
9         Q ← {s}
10        while Q ≠ 0
11             do u ← head[Q]
12                for each v ∈ Adj[u]
13                  do if color[v] = WHITE
14                     then color[v] ← GRAY
15                          d[v] ← d[u]+1
16                          π[v] ← u
17        ENQUEUE(Q,v)
18        DEQUEUE(Q)
19        color[u] ← BLACK
20   end
```

Figure 1. An overview of BFS algorithm according to (Cormen, 1992).

The main loop of the algorithm is contained in lines 9-18. The loop iterates as long as there remain gray vertices, which are discovered vertices that have not yet had their adjacency lists fully examined. Line 10 determines the gray vertex *u* at the head of the queue *Q*. The for-loop in lines 11-16 considers each vertex *v* in the adjacent list of *u*. If *v* is white, then it has not yet been discovered, and the algorithm discovers it by executing lines 13–16. It is first grayed, its distance *d[v]* is set to *d[u]+1*. Then, the vertex *u* is recorded as its parent. Finally, it is placed at the tail of queue Q. When all the vertices on *u'*s adjacency list have been explored, *u* is removed from Q and blacked in lines 17-18, where the color black, referred to a vertex, which had been discovered including all its adjacency list.

2.2.2 Depth-First Search Algorithm

Depth-first search processes the vertices first deep and then wide. DFS algorithm is illustrated in Figure 2, whereby the traversal algorithm visits all vertices $u \in V[G]$ reachable from an adjacent vertex before visiting another adjacent vertex. Lines 2-4 color all vertices white and initialize their π fields to NIL. Lines 5-7 check each vertex in V in turn and, when a white vertex is found, visit it using *DFS-VISIT*. Every time *DFS-VISIT* (u) is called in line 7, vertex *u* becomes the root of a new tree in the depth-first forest.
In each call *DFS-VISIT* (u), vertex u is initially white. Line 2 colors vertex *u* into gray, and lines 3 – 6 examine each vertex *v* adjacent to *u* and recursively visit *v* if it is white. As each vertex $v \in Adj[u]$ is considered in line 3, then, an edge *(u, v)* is explored by DFS. Finally, after every edge leaving u has been explored, line 7 paints u as a black.

```
DFS Algorithm (G)
1     {
2        for each vertex u ∈ V[G]
3            do color[u] ← WHITE
4                π[u] ← NIL
5        for each u ∈ V[G]
6            do  If (color[u] == white)
7                then DFS-VISIT (u);
8     }

DFS-VISIT (u)
1     {
2        color[u] ← GRAY
3        for each v ∈ Adj[u]
4            do if color[v] = white
5                then π[v] ← u
6                    DFS-VISIT (v);
7        color[u] = Black
8     }
```

Figure 2. An overview of DFS algorithm according to (Storer, 2002).

3. Water Distribution Network Discovery Tool

We have applied the foundation of computer science in utilizing a graph theory to model WDN and then, we have utilized both breath-first search (BFS) and depth-first search (DFS) algorithms to traverse the generated graph. The proposed WPN discovery tool facilitates the discovery of location of an unknown node from a selected root node. The tool explores the WDN graph utilizing the classical breath-first search algorithm separately and it displays the estimated distance between any two node-pairs during the traversals. We have coded the WPN discovery tool within Java Universal Network/Graph Framework (JUNG), which facilitates visual modeling of networks.

The snapshot of the WDN discovery tool is illustrated in Figure 3, which is an interactive tool providing choice to the user to design a typical WDN. The user can create a water distribution network by placing the nodes and the links in *EDITING* mode and user can move one or more nodes and links in *PICKING* mode. Hereby, WPN nodes are labeled in the order of generation. The button *TRANSFORMING* corresponds to move the entire graph to a specific area within the design space. The button, *ANNOTATING* marks the root node before traversal. The buttons, *Traverse BFS* and *Traverse DFS* correspond to the BFS and DFS traversal operations. The entire design area is cleared by *Clear* button. The buttons + and – perform the zoom operations.

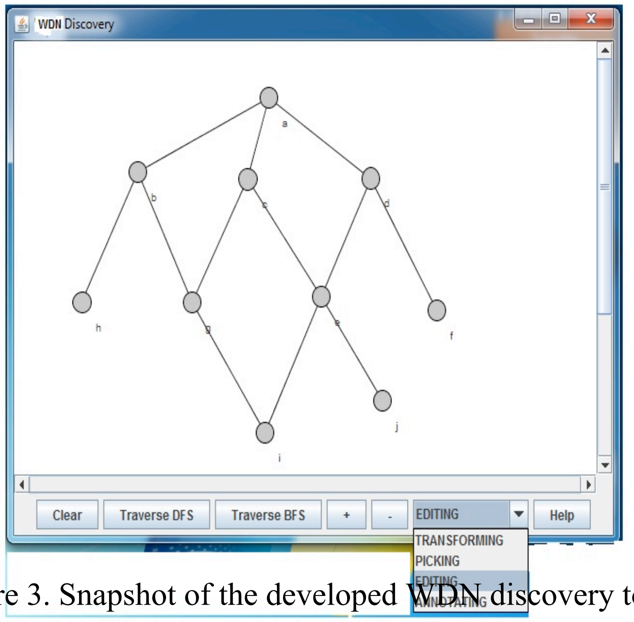

Figure 3. Snapshot of the developed WDN discovery tool.

4. Results and Discussions

We have considered a WDN comprising of pipes, tank, reservoir, pump and junctions, where we included loops and branches within WDN to reflect the structure of a practical distribution network within a street level. We have modeled the given WDN network as a graph with undirected edges and discovered the node locations using classical traversal algorithms.

4.1 Network Discovery Tool

The schematic diagram of a simple WDN is presented in Figure 4. The initial WDN graph generated using the WDN discovery tool is represented as in Figure 5. The snapshots of results after traversing the graph using DFS and BFS are given in Figures 6 and 7 respectively. In similarity with the data packet flooding in computer network discovery, our discovery tool stimulates a flooding of water from the selected root to discover its neighbors. The water flow is represented by the movement of red color line through the links. The initial nodes are presented as grey color circles and the selected root node is represented in yellow color. During the traversal, the visited nodes are represented as red color circles and the visited links are represented with thick lines.

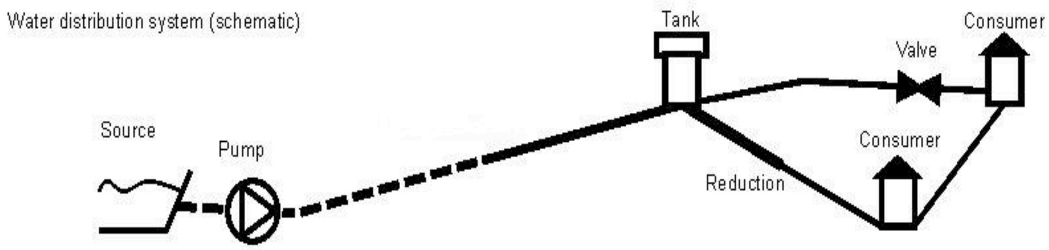

Figure 4. A Section of Water Distribution Network.

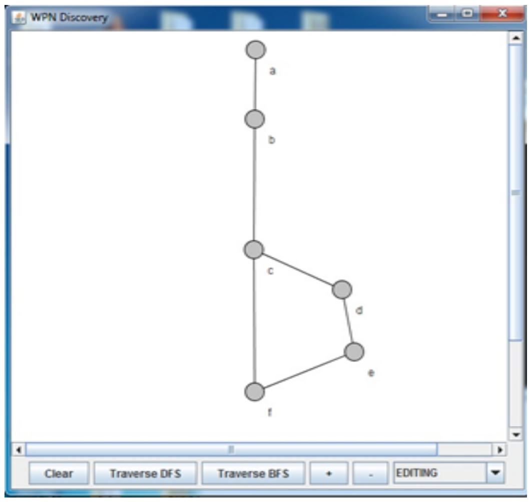

Figure 5. WPN discovery-initial network.

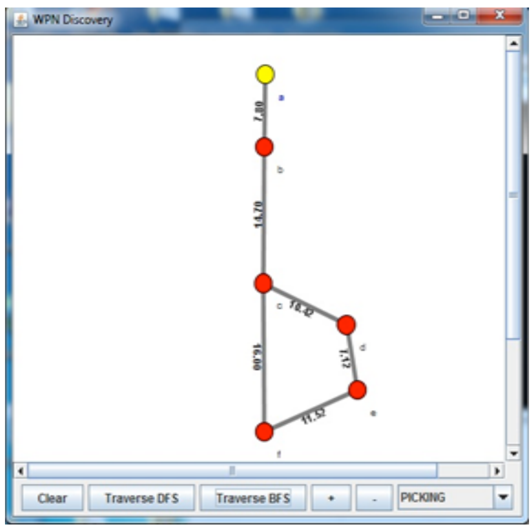

Figure 6. WPN discovery - DFS traversal.

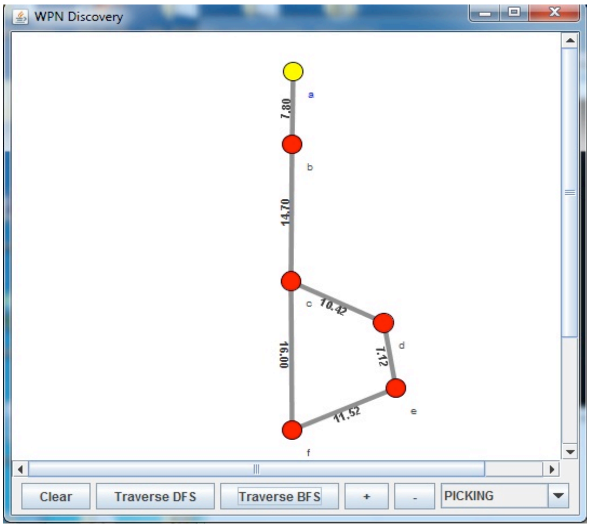

Figure 7. WPN discovery - BFS traversal.

In both DFS and BFS traversal, the Euclidian distance between two neighbor nodes are represented as the link weights. Our discovery tool has the flexibility in selecting any of the nodes as the root node within the generated graph.

5. Conclusion

We have developed a custom-made interactive network discovery tool utilizing Java Universal Network Graph (JUNG), which models the given WDN as a graph and it facilitates the discovery of location of unknown nodes from a selected root node by utilizing classical search algorithms, such as the breath-first and depth-first search algorithms. The outcome of the experiments utilizing WDN discovery tool will help us to design a low-cost technology hardware for water utility management in future.

References

AquaNET : http://www.inar.net/index.html

Balut, A., and Urbaniak, A. (2011). Management of Water Pipeline Networks Supported by Hydraulic Models and Information Systems, In the Proceedings of 12th International Conference on Carpathian Control, 25-28 May, Velke Karlovice, Czech Republic, 16-21.

Cormen, T.H., Leiserson, E., and Rivest, R.L. (1994). *Introduction to Algorithms*. McGraw-Hill, USA.

EPANET Hydraulic Analysis Package: *http://epanet.de/*

Frazey, J.S. (2004). The Development and Use of Distribution System Models for Improving the Security of Drinking Water System, Master Thesis. Airforce Institute of Technology, Ohio, USA.

Ghunaimi, Z. (2005). *Water Security in Kuwait and Arabian Gulf Countries.* published by Center for Research and Studies on Kuwait, Kuwait.

HydrauliCAD software*: http:// hydraulicad.com*

James A. S. (2002). *An Introduction to Data Structures and Algorithms*, Springer-verlag, NY, USA.

Madadhain, J., Fisher, D., Smyth, P., White, S., and Boey, Y.B. (2005). Analysis and Visualization of Network Data using JUNG. *Journal of Statistical Software,* pp. 1–25.

Paluszczyszyn, D., Skworcow, P., and Ulanicki, B. (2015). Modeling and Simulation of Water Distributed Systems with Quantized State System Methods. *Procedia Engineering,* 119, 554-563.

Prisco, G., D'Urso, M., Bernardi, G., Leopardi, A. and Schettino, F. (2011). An Effective Toolbox for Water Distribution Network Monitoring. In the Proceedings of the *5th European Conference on Antennas and Propagation*, 11-15 April, Rome, Italy, 886-887.

Rashid, S., Qaisar, S., Saeed, H., and Felemban, E. (2013). Performance Analysis of Wavelet Transforms for Leakage Detection in Long Range Pipeline Networks, In the Proceedings of *IEEE Conference on Systems, Process & Control*, 13-15 December, Kuala Lumpur, Malaysia.

Rossman, L.A. (2000). *EPANET 2 Users Manuel*. Water Supply and Water Resources Division, National Risk Managemnet Laboratory.

STANET software: *http://stafu.de/en*

Statistical Report on Water Network from Statistics Department and Information Center, Kuwait-2007-2011.

Yang, J., Li, Y., L., Zhang, N.F., Yang, J.F., Tang, J., and Su, Y.X. (2015). Water Quality Modeling for Pipeline Network Peripherals Based on Simulation of Water Age. In the Proceedings of *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications;* Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 26-28 October, Liverpool, UK, 2284-2290.

Contact email: sami.habib@ku.edu.kw