Computers and Chemical Engineering 108 (2018) 11–23

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

A graph partitioning algorithm for leak detection in water distribution networks

Aravind Rajeswaran[1], Sridharakumar Narasimhan*, Shankar Narasimhan

Department of Chemical Engineering, Indian Institute of Technology Madras, India

ARTICLE INFO

Article history:
Received 7 January 2017
Received in revised form 10 August 2017
Accepted 18 August 2017
Available online 30 August 2017

Keywords:
Water distribution networks
Leak detection
Graph partitioning
Multi-objective optimization

ABSTRACT

Urban water distribution networks (WDNs) are large scale complex systems with limited instrumentation. Due to aging and poor maintenance, significant loss of water can occur through leaks. We present a method for leak detection in WDNs using repeated water balance and minimal use of additional off-line flow measurements. A multi-stage graph partitioning approach is used to determine where the off-line flow measurements are to be made, with the objective of minimizing the measurement cost. The graph partitioning problem is formulated and solved as a multi-objective mixed integer linear program (MILP). We further derive an approximate method inspired by spectral graph bisection to solve the MILP, which is suitable for very large scale networks. The proposed methods are tested on large scale benchmark networks, and the results indicate that on average, flows in less than 3% of the pipes need to be measured to identify the leaky pipe or joint.

1. Introduction

It is estimated that energy generation accounts for approximately 15% of global water use, while approximately 8% of energy is used for treating and transporting water (Garcia and You, 2016). The resulting water-energy nexus is well studied and the process systems engineering community has contributed significantly in formulating and solving complex problems, e.g., in design (Rojas-Torres et al., 2014), operation (Shi and You, 2016), and monitoring for leaks (Mulholland et al., 2014; Kim et al., 2016) and contaminants (Palleti et al., 2016; Mann et al., 2012).

With the ever growing complexity and scale of water distribution networks (WDNs), leak detection is of particular importance for effective water management and quality control (Colombo and Karney, 2002; Puust et al., 2010). Leaky distribution systems are inefficient due to water loss, energy wastage, and unreliable water quality: especially in case of underground leaks. These effects are even more pronounced when the networks are poorly instrumented and maintained, such as in low and middle income countries.

In WDNs, leaks and losses are quantified using unaccounted-for water (UFW). High levels of UFW are detrimental to the financial viability of the system. These losses are a combined effect of real losses like leaks in pipes or joints, as well as other causes such as water thefts and unauthorized consumption (Gonzalez-Gomez et al., 2011). Given the growing concern towards uncertainty in quality water supplies, the problem of leak detection and control has grown in importance. Various techniques based on acoustic methods and magnetic flux leakage (Mpesha et al., 2001; Sun et al., 2011; Colombo et al., 2009) are available to determine the location of defect (either small defect like corrosion, or large leaks) in a single pipe. However, these methods can be time consuming, expensive, or disruptive in nature. Thus, it is beneficial to use these techniques only after narrowing down the leak to a small part of the network.

One approach to leak detection involves the use of hydraulic models and simulators. Available measurements are used to estimate the location of a leak which match the sensor measurements closely. This method is generally called inverse analysis (Liggett and Chen, 1994) and requires a large optimization problem to be solved. In order to use this approach, measurements of flow rates and pressures at a large number of intermediate locations are required, in addition to source pressure and demand flows. A more severe limitation of methods that make use of pressure measurements, is that the predictions depend on precise estimates of model parameters, including pipe friction factors, which are difficult to obtain. Practical applicability of this method to large scale networks have proven to be a hard task, as reported by some researchers (Stephens et al., 2004, 2005). A similar inverse problem in real time contaminant detection has been formulated that identifies the set of potential

* Corresponding author.
E-mail addresses: aravraj@cs.washington.edu (A. Rajeswaran), sridharkrn@iitm.ac.in (S. Narasimhan), naras@iitm.ac.in (S. Narasimhan).
[1] Currently at University of Washington, Seattle.

http://dx.doi.org/10.1016/j.compchemeng.2017.08.007
0098-1354/

Nomenclature

m	Number of edges in **G**
n	Number of vertices in **G**
$\mathbf{u}_1, \ldots, \mathbf{u}_n$	Eigenvectors of $\mathbf{J}\mathbf{J}^T$
w_k	Cost of edge k
$\mathbf{x} \in \{0, 1\}^n$	Indicator variable assigning each vertex to a partition
$\mathbf{z} \in \{-1, 1\}^n$	Indicator variable assigning each vertex to a partition
A	Adjacency matrix of **G**
D	Degree matrix of **G**
E	Set of edges
G(**N**, **E**)	Graph defined on **N** and **E**
J	Directed incidence matrix of **G**
L	Laplacian matrix of **G**
N	Set of vertices
$\mathbf{S}(\mathbf{N}_S, \mathbf{E}_S)$	Subgraph formed from **G**(**N**, **E**)
cut(**S**, $\bar{\mathbf{S}}$)	The cut-set of partition (**S**, $\bar{\mathbf{S}}$)
R(**S**, $\bar{\mathbf{S}}$)	The cut-cost of partition (**S**, $\bar{\mathbf{S}}$)
γ	Parameter used in goal programming formulation
$\lambda_1, \ldots, \lambda_n$	Eigenvalues of $\mathbf{J}\mathbf{J}^T$
μ	Weighting factor used in approximation formulation

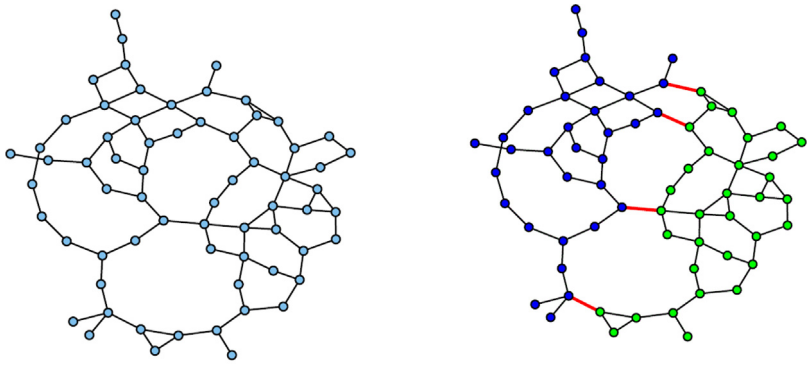
contamination locations (Mann et al., 2012). However, mixed integer linear programs (MILP) have to be solved in real time thus limiting this approach to networks with good communication and real time computational facilities.

Methods which do not use a hydraulic model explicitly in leak detection have also been proposed. One such method uses continuous pressure readings followed by filtering and statistical analysis to detect leaks, bursts and other abnormalities (Kim et al., 2016). On the other hand, Mulholland et al. (2014) describe a technique for leak detection using mass balances alone. The resulting system of equations is underdetermined and hence, a linear program with snapshots of data at different operating conditions is solved to obtain a most credible or plausible leak location. However, in the case of large networks, it is not clear if the leak can be localized to the desired degree of granularity. Furthermore, the method only exploits existing flow sensors and does not address the question of where to place additional flow sensors to address the problem of non-unique solutions and improve the leak resolution.

At present, most water networks are poorly instrumented with poor spatial and temporal resolution of data. In well instrumented networks, a small set of flow and pressure sensors are installed for the purpose of district metered area (DMA) sectorization, but these are few in number and consist of approximately 500–1000 connections. In addition, consumer consumption may be recorded over longer time scales while pressure changes are very rapid and occur over much shorter time scales. Thus, a leak if any, can be localized to a DMA or sub-network consisting of a large number of pipes or nodes, and further resolution of the leak is not trivial requiring significant time and effort.

In order to overcome the above difficulties, we propose a method for leak detection which uses only existing flow measurements, and some additional flow measurements which are repeatedly performed on-demand in field campaigns. We call this process of obtaining flow measurement in a pipe (on-demand) as **querying** the pipe for flow. Further, since the only property of leak we exploit is loss of material (water), the method is equally applicable to any form of loss including thefts. This method has the added advantage that it does not require knowledge of the pipe friction factors or pressure measurements.

To briefly illustrate the idea, consider the network shown in Fig. 1(a). Let us say that some node in this network is leaky, and our objective is to find it. By querying the red edges in Fig. 1(b), we can trace the leak to either of the two parts of the network, shown in blue and green. This is possible by exploiting water balance, or more broadly conservation laws, as will be shown in subsequent sections. By performing this operation repeatedly, we can arrive at a small part of the network which contains the leak. Querying a pipe requires access to it, which may be buried underground at a depth of about one to two meters. Hence, there is a non-negligible cost associated with every query. Therefore, it is important to minimize the number of queries, or query cost, which requires a strategic field campaign. An ideal field campaign should possess the following characteristics: (i) it should be systematic and arise out of a clear objective; (ii) it should scale to large sectors or the whole network in absence of DMAs; (iii) must be capable of assimilating information from other sources (like existing sensors); (iv) should be optimal, requiring only few queries and (v) should require minimal real time computations since communication and computational availability during a field campaign is likely to be limited. An algorithmic procedure for developing such a field campaign is the subject of this paper. We propose to solve this problem using a divide and conquer approach, where we divide the problem of leak detection using graph partitioning, and conquer the sub-problems using water balance. A high level description of the algorithm is shown in Fig. 2 which will be described in detail subsequently. $|S|$ is the size of sub-network with leak, and Th is a pre-defined threshold. The partitioning algorithm determines which pipes to measure at each stage, such that the network is partitioned into two sub-networks around each of which a flow balance can be applied. The proposed method



(a) Network with a leaky node

(b) Querying a set of pipes

Fig. 1. Illustration of querying the edges for flows and identifying the leaky part of the network.

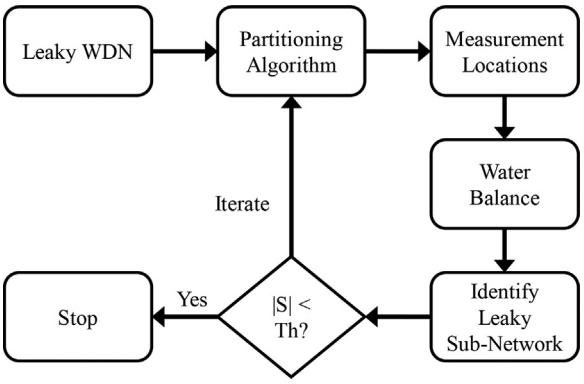


Fig. 2. Sequence of steps in proposed method.

is particularly suited for legacy networks with poor instrumentation, as well as for well instrumented networks since we make full use of existing sensors. Although we focus on WDNs in this work, the method itself is much more general and can be applied to any distribution system satisfying conservation laws.

2. Leak detection using graph partitioning

In this section, we first present the problem statement in the most general form. Next, we present an algorithm which describes our formulation and solution procedure. Finally, we also present an example to illustrate both the problem and the algorithm. The general objective of leak detection is to identify one or more leaky units, such as pipes and junctions. For most existing methods, in addition to existing permanent sensors, additional information like pressure readings in unmonitored parts of the network and system parameters like precise diameter and friction factor of pipes are required. Due to difficulties in obtaining some of these parameters and other reasons motivated in the Introduction, we explore a procedure which requires the use of only flow measurements which are measured on demand.

2.1. Assumptions

We first present the assumptions made in the formulation.

1. The WDN is in steady state condition.
2. The topology of the WDN (i.e., network connectivity) is known.
3. All supply (source) and demand (consumption) flow rates are measured continuously. No other permanent sensors are available.
4. We possess portable flow meters which can detect the flow rate as well as the flow direction. One example is ultrasonic flow meters which use time of flight principle.

Additionally, we make the following assumptions to simplify and aid the presentation. We will later describe modifications to our proposed approach to deal with situations when these assumptions are not satisfied.

1. There is only one leak in the network, and it is present in a node.
2. The sensors measurements do not contain any errors (either random or systematic).

2.2. Water balance for identifying leaks

Our proposed leak detection and localization algorithm makes use of flow rate measurements and flow balances around suitably selected subset of nodes of the WDN. When considering a generic

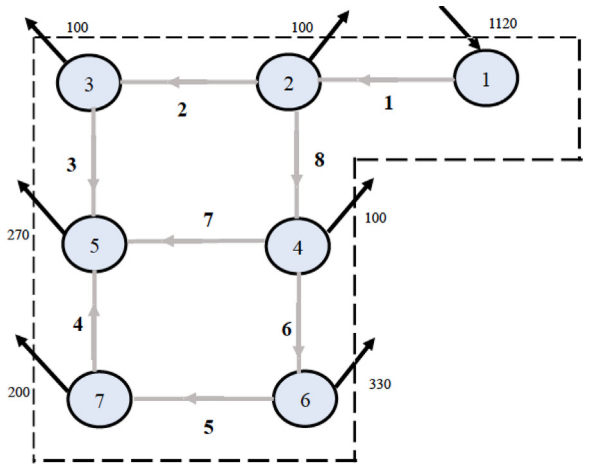


Fig. 3. Envelope around a network.

envelope encompassing a set of nodes and edges, it is possible to perform a water balance around this envelope. If there is no leak at any node within the envelope, then under steady state the flow balance around the envelope should be satisfied.

2.2.1. Example for water balance

The application of water balance is illustrated for an example in Fig. 3. In this figure, nodes numbered 1, 2, . . ., 7 are either supply or demand nodes, depending on the direction of the edges incident to or from the nodes indicated in Fig. 3. Flows in pipes denoted by edges labelled 1, 2, . . ., 8 are not measured.

Thus for this envelope, we can observe that the total inflow rate is 1120 units while the total demand is 1100 units. This discrepancy of 20 units indicates a leak of 20 units. Since the flows in the internal pipes are not measured, it is not possible to identify the source of the leak. We propose to localize the leak by probing unmonitored pipes to determine the flow rate and direction and carrying out an appropriate balance. Clearly, in order to apply the flow balance to an envelope, the flow rates in the pipes crossing the envelope have to be measured. The flow directions in these pipes are also required, for which we need appropriate instrumentation. If such instrumentation is not available, alternate methods to ascertain flow direction must be used. One approach could be to use nominal case hydraulic simulations, and assume the flow directions do not change in presence of leak, which is likely valid if the magnitude of leak is small. In general, we assume that either appropriate instrumentation or alternate methods to find flow directions are available.

The challenge is to select a suitable set of pipes for measurement at each stage so that the network is partitioned into two sub-networks allowing a water balance to be carried out on both the sub-networks. E.g., in Fig. 4, selecting pipe 2 does not partition the network in the desired manner. On the other hand, measuring pipe 1 isolates node 1 from the rest of the network and partitions it into sub-networks, one containing node 1 and the other containing nodes 2, 3, . . ., 8. Hence, a balance can be written across these sub-networks and it is possible to isolate the leak to one of the sub-networks. However, the resulting sub-networks are imbalanced, in that one sub-network contains one node and the other 7. On the other hand, measuring flows across pipes 2, 7 and 5 partitions the network into almost balanced partitions. However, it is possible to obtain similar balanced partitions using fewer number of measurements, e.g., by measuring flows in pipes 3 and 8 resulting in partitions shown in Fig. 4(a) and (b). The flows in pipes 3 and 8 are 120 and 800 units with directions as shown in Fig. 4. In order to maintain consistency, two artificial nodes M_1 and M_2 are introduced to denote the sensors. In the partition shown in Fig. 4(a), M_1

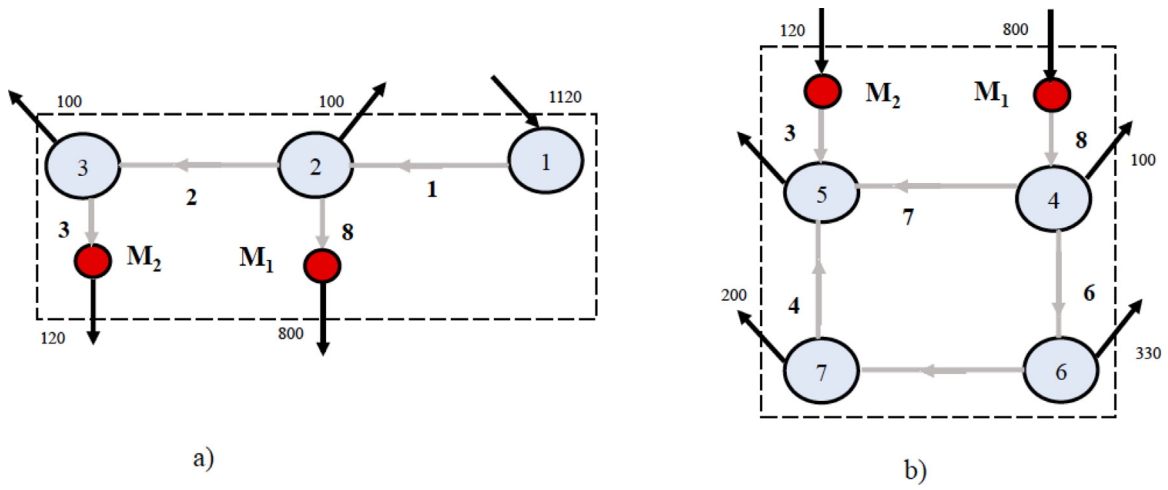


Fig. 4. Partitioning the network: Stage 1.

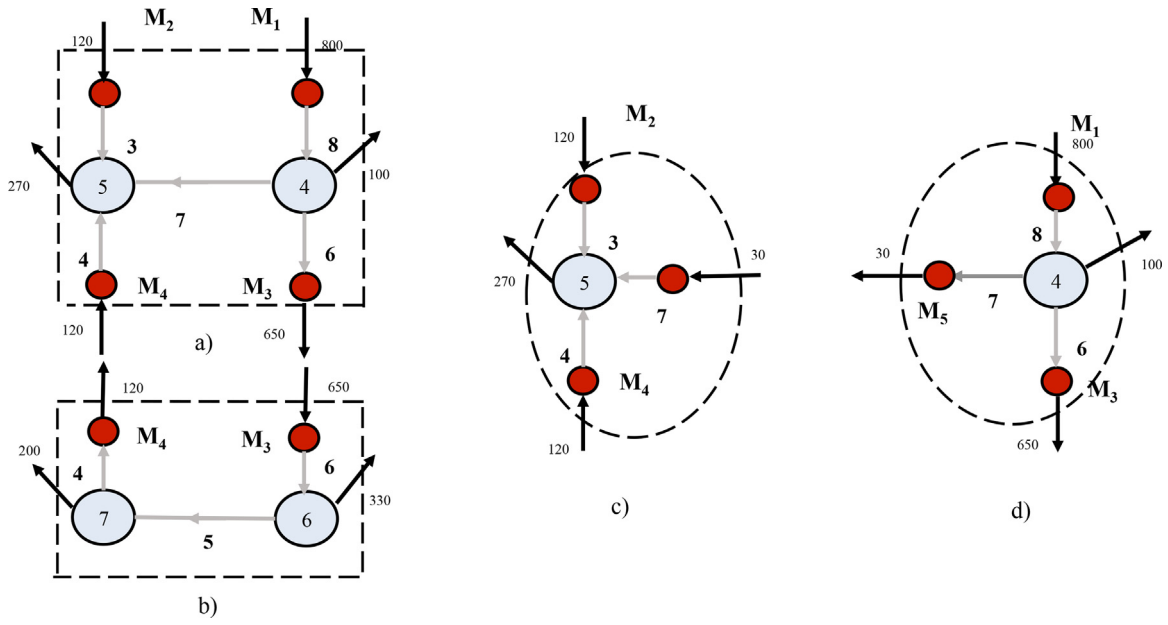


Fig. 5. Partitioning the network: Stages 2 and 3.

and M_2 act as demand nodes with flow rates of 120 and 800 units respectively. Conversely, in Fig. 4(b), M_1 and M_2 act as supply nodes. Applying a water balance around the envelope shown in Fig. 4(a) do not indicate any leak. A similar balance around Fig. 4(b) indicates a deficit of 20 units indicating that a leak can be attributed to one of the nodes 4, 5, 6 or 7.

In the next stage, we probe pipes 4 and 6 partitioning the network into two sub-networks shown in Fig. 5(a) and (b). Sensors in pipes 4 and 6 are represented by pseudo nodes M_4 and M_3 respectively. Flow rates in pipes 4 and 6 are respectively 120 and 600 with directions as shown. Flow balance on the partition in Fig. 5(a) reveals an imbalance of 20 units localizing the leak to either node 5 or 4. In the third and final stage, pipe 7 is probed resulting in the sub-networks shown in Fig. 5(c) and (d). Water balance over node 4 (Fig. 5(d)) reveals conclusively that the leak is in node 4.

2.3. Algorithm for leak detection

As described in the preceding example, the leaky unit can be identified using only a small set of measurements chosen carefully in sequence. This can be formalized as a multi-stage graph partitioning problem. The basic idea is to decompose the network into smaller sub-networks, and trace the leak to one of these sub-networks. The important step in this divide and conquer procedure is to attribute the leak to one of the sub-networks, which is possible through water balance. In doing so, at each step we incur a cost $R(\mathbf{S}, \bar{\mathrm{S}})$, associated with this step. The goal is to find a partitioning algorithm that will minimize the total cost. Since the leak location is not known *a priori*, this proves to be a very difficult problem. At any stage, we will not know which subgraph contains the leak until the partition is actually made and the corresponding cost incurred. Thus, the number of possibilities which we need to search over is exponentially large. To overcome this, we provide an approximate approach or heuristic where at each iteration in the algorithm, we minimize only $R(\mathbf{S}, \bar{\mathrm{S}})$ subject to some constraints which indirectly measures cumulative cost in subsequent iterations. A sketch of this procedure is presented in Algorithm 1.

Algorithm 1. Leak detection algorithm.

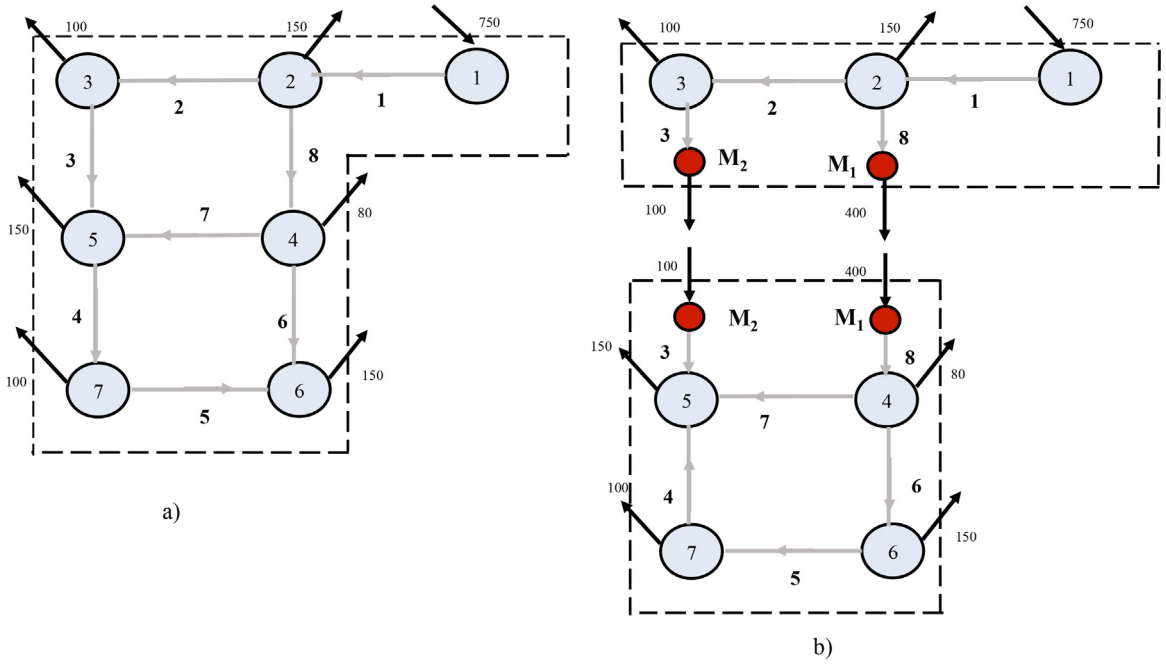


Fig. 6. Leak detection using partitioning: different loading conditions.

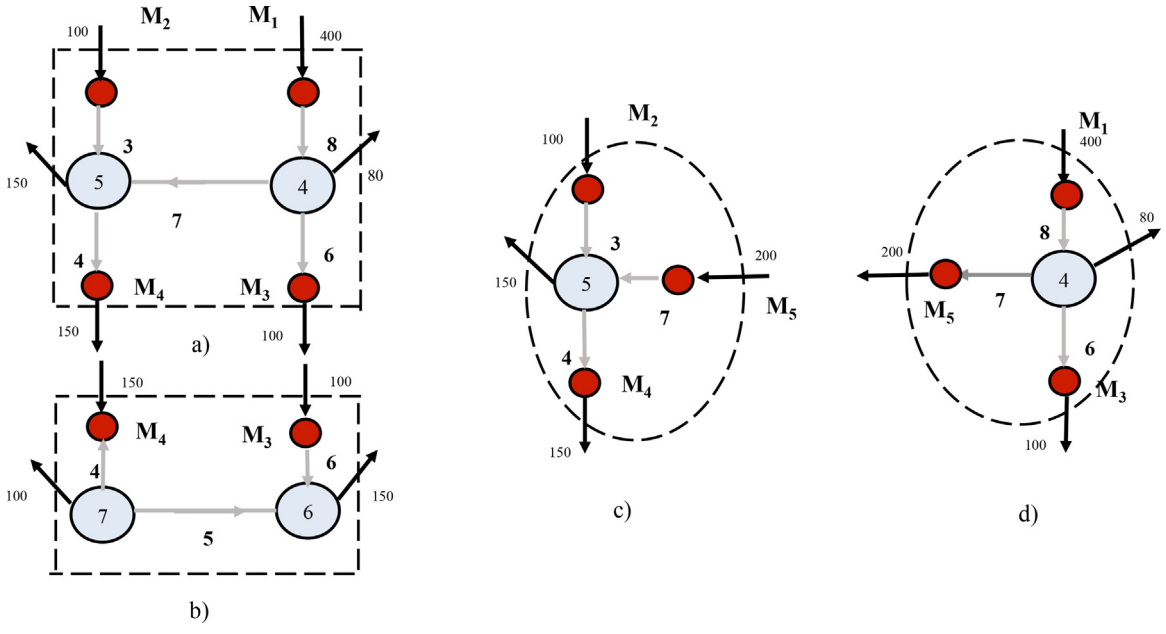


Fig. 7. Leak detection using partitioning under different loading conditions: Stages 2 and 3.

Algorithm 1: Leak detection algorithm

Input: Graph $\mathbf{L}(\mathbf{N}, \mathbf{E})$ containing leaky node, δ (threshold)
Initialize: $\mathbf{G} \leftarrow \mathbf{L}$; Cost $\leftarrow 0$
while *size*$(\mathbf{G}) > \delta$ **do**
  $(\mathbf{S}, \bar{\mathbf{S}}) \leftarrow$ partition $(\mathbf{G})$
  $\mathbf{G} \leftarrow$ find leaky partition $(\mathbf{S}, \bar{\mathbf{S}})$
  Cost $\leftarrow$ Cost $+ R(\mathbf{S}, \bar{\mathbf{S}})$
end
Result: Leaky node is in vertex set of $\mathbf{G}$

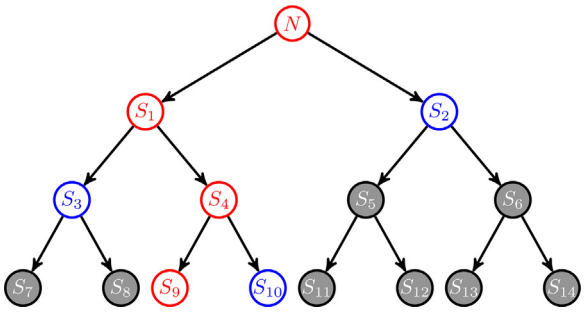


Fig. 8. Decision tree for leak detection.

It must be noted that the partitioning strategy does not depend on the flow rates or operating conditions and only depends on the leak location, network topology and the cut-cost. E.g., consider the network shown in Fig. 6(a) which is identical topologically to the one in Fig. 3, but operating under different conditions. A leak of 20 units is immediately observable from an overall flow balance. The subsequent partitioning strategy is the same: probe pipes 3 and 8 (as shown in Fig. 6(b). Further measurements and partitions in the second and third stages are shown in Fig. 7(a)–(d). Hence, it is clear, that if the leak is in node 4, irrespective of the operating conditions, the partitioning and measurement strategy does not change.

The biggest advantage of a stage wise approach is the ability to pre-compute the necessary partitions off-line, which can be stored in the form of a decision tree. Such a decision tree, as shown for example in Fig. 8, provides a compact representation for the sequence of operations to be followed. The nodes in this decision tree represent induced subgraphs of the original WDN. The figure illustrates the path traced for one particular instance of leak in the WDN. Nodes in red are sub-graphs that contain the leak, nodes in blue are sub-graphs that have been verified to be balanced (i.e. does not contain a leak) and grey nodes are unexplored sub-graphs. In the event of a leak, the task of where and how to make measurements reduces to a look-up exercise. Such a representation is also immediately useful to field engineers and serve as a decision support or monitoring system. For example irrespective of the leak location, at the start of the process, the pipes between S_1 and S_2 has to be probed first. Hence in networks where leaks and thefts occur frequently, it might be desirable to monitor these pipes closely, possibly even add permanent sensors. Since the performance of this procedure is directly linked to our ability to find good partitions, we explore various partitioning algorithms in Sections 3 and 4.

2.4. Leaks in pipes

In the main text, we presented the algorithm for finding leaks when they occur in nodes. However, in some cases, leaks may occur at any point along pipes as well. We now present an extension of the method for this case. We continue with the same assumptions except that rather than assuming leak in a node, we assume that the leak is in a pipe.

Consider the graph **G** which contains the leak (either the full network, or network under consideration in some step of the recursive procedure). We consider a possible partition into **S** and $\bar{\mathbf{S}}$ by querying flows in cut(**S**, $\bar{\mathbf{S}}$). For the above scenario, a straightforward approach to querying a pipe is to measure the flows at both its end points – very close to the node, as shown in Fig. 9). If the flow at M_{1a} and M_{1b} do not match, it is clear the leak is in the pipe. However, if the flow rates at M_{1a} and M_{1b} are equal, then the leak is definitely not in this pipe. Following a similar procedure for all the pipes in cut(**S**, $\bar{\mathbf{S}}$), we can trace the leak to either **S** or $\bar{\mathbf{S}}$ exactly. In other words, the leaky node or pipe is within the partition. In this strategy, the cost will be twice the cut-cost.

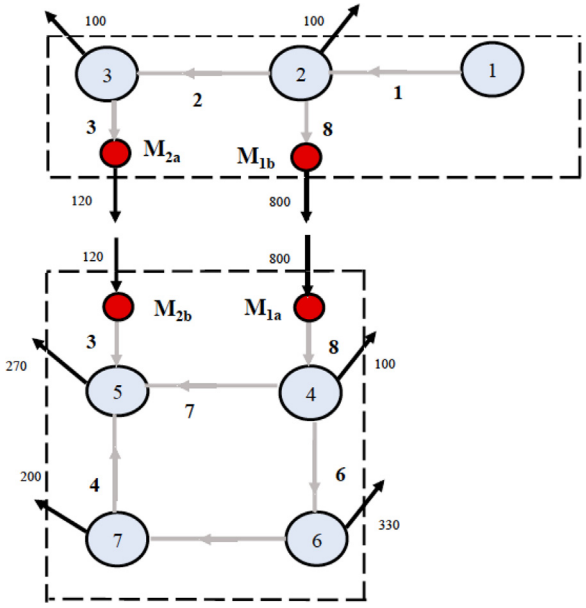


Fig. 9. Leaks in pipes.

However, it is possible to reduce the cut-cost with some modifications. When querying a pipe for the first time, instead of making two measurements, we can measure the flow at a single point close to the center. In this case however, the leak need not be in the interior of either partition. Since leaks can occur at any point on a pipe, the half-pipe segments of the crossing pipes (part of cut-set) could contain the leak. Thus we modify our definition of partition to include these pipe segments as well (which are incident on only one node). We do this by introducing an *artificial* node at the point of measurement. Thus an edge between an actual node and artificial node represents a pipe segment. With this modification, the recursive procedure proposed in the preceding section can be used. After many recursion steps, we may come to a stage where we need to query an edge between an actual node and artificial node. This amounts to measuring a pipe for the second time, where we already have one measurement for the pipe. In such a case, the second measurement is made close to the actual node. If this measurement does not match with the flow measurement obtained at the artificial node, then the pipe segment contains the leak and the process can be stopped. When the measurements match, we continue with the recursive procedure by eliminating the pipe segment. In this procedure, only a few pipes will be measured twice and, therefore, the cumulative number of measurements required will be fewer.

3. Graph partitioning – MILP formulation

In this section, we formulate the graph partitioning problem as an optimization problem and provide various mixed integer linear programming (MILP) formulations for the same. We define an indicator vector $\mathbf{x} \in \{0,1\}^n$ that assigns each node to a partition. For instance, $x_i = 1$ implies node $i \in \mathbf{S}$ and $x_i = 0$ implies node $i \in \bar{\mathbf{S}}$. We restrict our attention to the non-trivial case of a connected graph. Let the water network be described by a graph G(V,E) and let J be the directed incidence matrix of G. Consider the kth element of $\mathbf{J}^T\mathbf{x}$, where edge k connects nodes i and j. This can be simplified as:

$$\sum_{l=1}^{l=n} J_{lk} \quad x_l = J_{ik}x_i + J_{jk}x_j, \tag{1}$$

since only two rows in the kth column of $\mathbf{J}$ are non zero (by construction). Further, assuming without loss of generality that $i<j$, we have $J_{ik}x_i + J_{jk}x_j = x_i - x_j$. We further make the following observation:

$$\sum_{l=1}^{l=n} J_{lk}x_l = x_i - x_j = \begin{cases} 0 & x_i = 1, x_j = 1 \quad \text{or} \quad x_i = 0, x_j = 0 \\ 1 & x_i = 1 \quad \& \quad x_j = 0 \\ -1 & x_i = 0 \quad \& \quad x_j = 1 \end{cases} \tag{2}$$

which indicates that the cut-size is equal to the number of non-zero entries in $\mathbf{J}^T\mathbf{x}$. This can be related to the cut-cost $R(\mathbf{S}, \bar{\mathbf{S}})$ through:

$$R(\mathbf{S}, \bar{\mathbf{S}}) = \sum_{k=1}^{k=m} w_k \left| \sum_{l=1}^{l=n} J_{lk} \; x_l \right| \tag{3}$$

where w_k is the cost of querying edge k. When presenting the results, we set $w_k = 1 \;\; \forall k$, but it is clear that the method naturally extends to arbitrary values. As a proxy for minimizing the total cumulative cost, we propose to minimize $R(\mathbf{S}, \bar{\mathbf{S}})$ while ensuring that the partitions are balanced. This constraint is important to avoid partitions that myopically reduce $R(\mathbf{S}, \bar{\mathbf{S}})$ to provide lopsided partitions which may require large cut-costs in subsequent iterations. Parallels can be drawn with binary search, except that in our problem cost for different splits are different. The two objectives of minimizing $R(\mathbf{S}, \bar{\mathbf{S}})$ and keeping the partition balanced are likely to be conflicting. Thus, multi-objective optimization is the most general way to pose this problem, as presented below:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \left| n - 2 \times \sum_{i=1}^{i=n} x_i \right| \text{ and } \sum_{k=1}^{k=m} w_k \left| \sum_{l=1}^{l=n} J_{lk} \; x_l \right| \\ \text{s.t.} \quad & \mathbf{x} \in \{0,1\}^n \end{aligned} \tag{4}$$

We first propose to remove the absolute expressions so that we can formulate the problem as a standard MILP. Consider the constraint: $\sum_{i=1}^{i=n} x_i \leq \frac{n}{2}$, which serves dual purposes. Firstly, it removes the absolute value expression in the first objective. Additionally, it helps in pruning the search space by removing symmetries. For any feasible solution $\mathbf{x}$ the solution $\mathbf{1} - \mathbf{x}$ is equally valid and would produce the same value for the objective function. By introducing the above constraint, this symmetry is broken. In order to remove the absolute value expression in the second objective, we introduce two new m-vectors ($\mathbf{t}_1$ and $\mathbf{t}_2$) as decision variables with the following conditions:

$$\mathbf{t_1} - \mathbf{t_2} = \mathbf{J}^{\mathrm{T}}\mathbf{x}$$

$$\mathbf{t_1} + \mathbf{t_2} \leq \mathbf{1}$$

$$\mathbf{t_1} \in [0,1]^m \; \mathbf{t_2} \in [0,1]^m$$

Minimizing the second objective is now equivalent to minimizing $\sum_{k=1}^{m} t_{1k} + t_{2k}$, since an element of the vectors, t_{1k} or t_{2k} take the value 1 only when $x_i - x_j = \pm 1$, and are forced to 0 (minimization) whenever $x_i - x_j = 0$. Hence the optimization problem can now be written as:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{t_1},\mathbf{t_2}} \quad & \overbrace{\left(-2 \times \mathbf{1}^{\mathrm{T}}\mathbf{x}\right)}^{\text{Size Disparity}} \text{ and } \overbrace{\left(\mathbf{w}^{\mathrm{T}}\mathbf{t_1} + \mathbf{w}^{\mathrm{T}}\mathbf{t_2}\right)}^{\text{Cut-Cost}} \\ \text{s.t.} \quad & \mathbf{t_1} - \mathbf{t_2} = \mathbf{J}^{\mathrm{T}}\mathbf{x} \\ & \mathbf{t_1} + \mathbf{t_2} \leq \mathbf{1} \\ & \mathbf{1}^{\mathrm{T}}\mathbf{x} \leq 0.5n \\ & \mathbf{x} \in \{0,1\}^n \; \mathbf{t_1} \in [0,1]^m \; \mathbf{t_2} \in [0,1]^m \end{aligned} \tag{5}$$

There are several methods to solve multi-objective optimization problems, which are based on some notion of relative importance of the different objectives. One simple method is to scalarize the objective function by assigning relative weights to the different objective functions. In this case however, there is no obvious metric to trade off one for the other. One popular method in computer vision literature is normalized cuts (Shi and Malik, 2000), which proposes a ratio measure. However, in our problem, the end goal is to minimize total cost, for which the n-cut scalarization does not have a clear physical interpretation. We study two different paradigms which provide better physical insight for this application.

3.1. Lexicographic solution

The first method is based on lexicographic optimization where one objective is given infinite precedence over the other. Lexicographic solutions to multi-objective optimization has been extensively researched (Sherali and Soyster, 1983) and also successfully applied to sensor placement problems (Bhushan and Rengaswamy, 2002; Bhushan et al., 2008). In this paradigm, the objective with maximum precedence is minimized first and among multiple solutions which can achieve this, the one which minimizes the second objective is picked, and so on. In this problem, we give more importance to the balanced partitioning objective, and admit only those solutions which produce equally balanced partitions. Conceptually, among the $\binom{n}{n/2}$ possible solutions for obtaining balanced partitions, the solution which minimizes the cut-cost is chosen. However, we do not explicitly enumerate all these possibilities, but instead solve the following MILP.

$$\begin{aligned} \min_{\mathbf{x},\mathbf{t_1},\mathbf{t_2}} \quad & \mathbf{w}^{\mathrm{T}}\mathbf{t_1} + \mathbf{w}^{\mathrm{T}}\mathbf{t_2} \\ \text{s.t.} \quad & \mathbf{t_1} - \mathbf{t_2} = \mathbf{J}^{\mathrm{T}}\mathbf{x} \\ & \mathbf{t_1} + \mathbf{t_2} \leq \mathbf{1} \\ & \sum_{i=1}^{i=n} x_i = \lfloor \frac{n}{2} \rfloor \\ & \mathbf{x} \in \{0,1\}^n \; \mathbf{t_1} \in [0,1]^m \; \mathbf{t_2} \in [0,1]^m \end{aligned} \tag{6}$$

3.2. Goal programming

The second method is goal programming, where we set a nominal goal for one objective. For example, this can be introduced in the form of a constraint, so that the search space is confined to those situations which meet this goal. We use this idea for our problem to get a good handle over the partition sizes. By deviating a little from exact bisection, we may be able to reduce the cut-cost significantly. In such cases goal programming can be very effective. We assign a goal on the partition size to take the form $\sum_{i=1}^{n} x_i \geq (0.5 - \gamma)n$ which guarantees a minimum size for both partitions. γ is a parameter which defines the level of goal on the partition sizes. Based on our simulations, a good choice is $\gamma = 0.1$, so that partitions have a minimum size of $0.4n$.

$$\begin{aligned} \min_{\mathbf{x},\mathbf{t_1},\mathbf{t_2}} \quad & \mathbf{w}^{\mathrm{T}}\mathbf{t_1} + \mathbf{w}^{\mathrm{T}}\mathbf{t_2} + \left(-\frac{\epsilon}{n}\mathbf{1}^{\mathrm{T}}\mathbf{x}\right) \\ \text{s.t.} \quad & \mathbf{t_1} - \mathbf{t_2} = \mathbf{J}^{\mathrm{T}}\mathbf{x} \\ & \mathbf{t_1} + \mathbf{t_2} \leq \mathbf{1} \\ & \sum_{i=1}^{i=n} x_i \leq \lfloor \frac{n}{2} \rfloor \\ & \sum_{i=1}^{i=n} x_i \geq \lceil (\frac{1}{2} - \gamma)n \rceil \\ & \mathbf{x} \in \{0,1\}^n \; \mathbf{t_1} \in [0,1]^m \; \mathbf{t_2} \in [0,1]^m \end{aligned} \tag{7}$$

Here we have added an additional term to the objective function, $(\epsilon/n)\mathbf{1}^T\mathbf{x}$. By choosing ϵ appropriately, we can have $(\epsilon/n)\mathbf{1}^T\mathbf{x} < min(\mathbf{w})$ and hence less than the minimum possible change in $(\mathbf{w}^T\mathbf{t}_1 + \mathbf{w}^T\mathbf{t}_2)$. Thus, addition of this term cannot alter the optimal value of cut-cost. The purpose of this term is to ensure that if there are multiple minimum cut-cost solutions which meet the partition size goal, we would obtain the most balanced partition.

4. Graph partitioning – approximation

The general problem of graph partitioning with partition size or cardinality constraints are NP hard (Arora et al., 2009). For large problems, it may be computationally demanding to solve the above ILP problem and obtain the optimal solution. Hence we discuss some approximate solution methods.

The idea of approximation algorithms for ILPs involve two steps: relaxing some constraints to solve a simpler problem, and a rounding-off step where solutions consistent with the actual constraints are recovered from the relaxed solutions. Graph partitioning has many approximation algorithms in literature which have been successfully used in different domains. For our application, approximation algorithms have two uses: it can be used to find quick and reliable estimates of upper-bound associated with the field campaign, thereby help make informed policy calls regarding the feasibility of the field campaign. Additionally, it can also be used for the first few levels of very large networks, where ILPs become computationally expensive. For the subsequent levels, when network size has reduced greatly, the ILP algorithm can be used.

The relaxation step is both problem and application specific. For example, if one clearly knows the sizes of the partition – a scenario common in circuit design where number of components to be placed on a chip is known, a popular method of choice is the Kernighan and Lin algorithm (Kernighan and Lin, 1970). This method is not applicable to our problem since cut-cost is directly tied to our overall objective, and partition sizes cannot be accurately predicted. Under such circumstances, methods from spectral graph theory are more appropriate. Our approach in spirit follows from the goal programming ILP formulation, and we ultimately arrive at a result which is similar to the spectral bisection method (Pothen et al., 1990) but with subtle differences and alternative interpretations.

Consider an assignment variable for nodes to different partitions chosen as $\mathbf{z} \in \{-1, 1\}^n$ which is equivalent to the earlier choice of $\mathbf{x} \in \{0, 1\}^n$ through the transformation $\mathbf{z} = (2\mathbf{x} - \mathbf{1})$. We again wish to arrive at an assignment that minimizes cut-cost subject to partition size constraints.

$$R(\mathbf{S}, \bar{\mathbf{S}}) = \frac{1}{2}\|\mathbf{J}^T\mathbf{z}\|_1 = \frac{1}{4}\|\mathbf{J}^T\mathbf{z}\|_2^2$$

where $\|.\|_1$ and $\|.\|_2$ represent the 1-norm and 2-norm respectively. For cases where edges have different costs, we can use the simple modification: $\{J_{ik} = w_k, J_{jk} = -w_k\} \forall k$ such that edge k connects nodes i and j. In the previous section, the objective was formulated using the 1-norm. We now choose to minimize the 2-norm to obtain an approximate analytical solution. Assigning some relative cost μ (unknown) to the two objectives, the problem can be posed as:

$$\begin{aligned} \min_{z} \quad & \underbrace{\left(\mathbf{z}^T\mathbf{J}\mathbf{J}^T\mathbf{z}\right)}_{\text{Cut-Cost}} + \underbrace{\mu\left(\mathbf{1}^T\mathbf{z}\right)^2}_{\text{Size Disparity}} \\ \text{s.t.} \quad & \mathbf{z} \in \{-1, 1\}^n \end{aligned} \tag{8}$$

The goal programming size constraint will be imposed explicitly at a later stage. We relax the integral constraint on $\mathbf{z}$, viz., $\mathbf{z} \in \{-1, 1\}^n$ to $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{z}^T\mathbf{z} = n$. Next we express $\mathbf{z}$ using the orthonormal set of eigenvectors of $\mathbf{J}\mathbf{J}^T$. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues

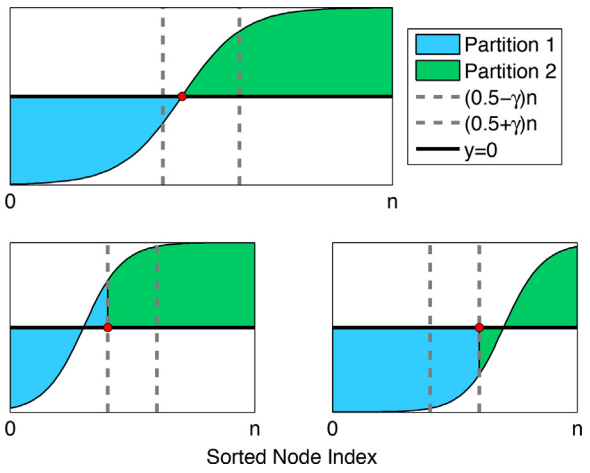


Fig. 10. Incorporating size restriction to approximation algorithm.

of $\mathbf{J}\mathbf{J}^T$ sorted in ascending order with eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ respectively. Defining $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$ we can write $\mathbf{z} = \mathbf{U}\boldsymbol{\alpha}$ where $\boldsymbol{\alpha}$ is the vector of projections onto $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$. As shown in the appendix, we have $\mathbf{L} = \mathbf{J}\mathbf{J}^T$, where $\mathbf{L}$ is the Laplacian matrix. From Properties 2 and 3 of the Laplacian matrix, we have $\lambda_1 = 0$, $\mathrm{u}_1 = \frac{\mathbf{1}}{\sqrt{n}}$ and $\lambda_i > 0$, $i = 2, \ldots, n$. The constraint $\mathbf{z}^T\mathbf{z} = n$ is equivalent to $\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_n^2 = n$ With this change of variables, the optimization problem in (6) after relaxation becomes:

$$\begin{aligned} \min_{\alpha} \quad & \alpha_2^2\lambda_2 + \alpha_3^2\lambda_3 + \cdots + \alpha_n^2\lambda_n + n\mu\alpha_1^2 \\ \text{s.t.} \quad & \alpha_1^2 + \alpha_2^2 + \cdots + \alpha_n^2 = n \end{aligned} \tag{9}$$

with $\lambda_2 \leq \lambda_3 \cdots \leq \lambda_n$. The above problem can be solved analytically as follows:

1. If $n\mu \leq \lambda_2$, then $\alpha_1^2 = n$ and $\alpha_i = 0\ \forall i \neq 1$
2. If $n\mu > \lambda_2$, then $\alpha_2^2 = n$ and $\alpha_i = 0\ \forall i \neq 2$

The first solution indicates that if cut-cost is significantly more than cost associated with size disparity in partitions, the obvious solution is to not to partition the graph at all. This solution is trivial and is discarded. The second solution indicates that if cost associated with disparity is more than a certain threshold, then the solution is to partition such that $\alpha_2^2 = n$. This suggests the assignment choice as $\mathbf{z} = \sqrt{n}\mathrm{u}_2$ where $\mathbf{u}_2$ is the eigenvector corresponding to the second smallest eigenvalue, also known as the Fiedler vector. Since $\mathbf{u}_2 \neq \mathbf{0}$ is orthogonal to $\mathrm{u}_1 = \frac{\mathbf{1}}{\sqrt{n}}$, $\mathbf{u}_2$ is non-trivial. In order to obtain an integer solution, we employ a simple round off procedure to obtain the solution $\mathbf{z}$ that is consistent with problem specifications, and also maximizes α_2^2. The final solution is:

$$\mathbf{z} = \mathrm{sgn}(\mathrm{u}_2)$$

Note that the above solution maximizes α_2^2 which is only an approximation of the original problem. In order to minimize the true problem (6), we need to consider the relative magnitudes of the different eigenvalues which is possible only in a combinatorial setting. In fact, it is this approximation that enables us to arrive at a computationally tractable solution.

Partitioning based on entries of Fiedler vector is known by the name of spectral bisection (Pothen et al., 1990) and is known to produce skewed partitions (Shi and Malik, 2000). This problem can be tackled by explicitly imposing a goal programming constraint as shown in Fig. 10. We sort the entries of $\mathbf{u}_2$ in ascending order, and normally assign partitions based on sign of the entry corresponding to each node. If we get skewed partitions, we can cut-off the partitions at the threshold defined by the minimum partition sizes.

Table 1
Properties of the networks studied. (n and m are the number of nodes and edges respectively; q is the link density ($\frac{2m}{n(n-1)}$); $\langle k \rangle$ and k_{max} are the mean and maximum node degrees).

Network	n	m	q	$\langle k \rangle$	$k_{\max}$
Exnet	1893	2418	1.35e−3	2.55	10
CO. Springs	1786	1992	1.25e−3	2.23	4
Richmond	872	957	2.52e−3	2.20	4
Dtown	401	459	5.72e−3	2.29	5
Bangalore	150	155	1.43e−2	2.07	5

This is shown schematically in Fig. 10. This assignment ensures that α_2^2 is maximized when adhering to the partition size constraints. This is because there is a fixed number of sign mismatches that would occur between z_i and $u_2(i)$ which reduces the value of α_2 from its maximum possible value. By sorting and assigning nodes to partitions such that sign mismatches always occur with $u_2(i)$ of least magnitude, the maximum possible value of α_2 is achieved in presence of the partition size constraint.

Remark. While we have presented two methods here (ILP and approximation scheme) which work well for the target application as seen through case studies, researchers have attempted other approximation algorithms, and the field of graph partitioning is very rich in literature. Some of these methods employ the use of semi-definite programming and randomized algorithms (Guruswami and Sinop, 2011; Arora et al., 2009). We do not present the results of these algorithms since the size of benchmark networks considered in this paper were not large enough to render ILPs computationally intractable, and the spectral bisection method provides adequate performance. However, if necessary, it is trivial to incorporate other approximate partitioning methods in the algorithm.

5. Results and discussion

To test the proposed methods, we have chosen representative water distribution networks used frequently in literature. These include the EXNET, Richmond, DTown, and Colorado Springs networks. Yazdani and Jeffrey (2011) have studied the topology of these networks, where they analyze properties like link density, clustering coefficient, betweenness centrality, etc.

We have chosen these networks due to the wide spectrum of size, formation, and organizational patterns; and hence representative of most WDNs (Yazdani and Jeffrey, 2011). The EXNET network is a large realistic benchmark problem used for multi-objective optimization of water systems. The Colorado Springs network is an example with multiple water supply sources, while the Richmond network is a sub-network of the Yorkshire Water system in the UK with a single reservoir. The DTown network was used in the Battle of the Water Network II (BWN-II) as a design problem. In addition, we have also tested the algorithm on one sector of the Bangalore water distribution network, which is smaller in size compared to the other "full" networks, to study how the methods perform at smaller scales. Some important properties of these networks are summarized in Table 1. The layouts of these networks are illustrated in Fig. 11.

5.1. Case study 1 – leak in nodes

In the first case study, we assume that leaks are always present in nodes. This is a continuation of the assumptions made in the problem formulation. We choose to apply the algorithm repeatedly till we find the leaky node. We simulate the leak in a node, and apply the algorithm till we identify it, and record the number of additional flow measurements (other than the source and demand flow rates) required. This simulation procedure is repeated with a leak simulated in every possible node in turn. This corresponds to a full enumerative study since we iterate over every possible leak scenario. It should be noted that in the simulation, the flow rate values are unimportant (and are therefore not reported), since we have assumed that these measurements are noise free. Consequently, the flow balance applied to every partition not containing a leak will be exactly satisfied, and only the flow balance corresponding to the partition containing the leaky node will have a residual. The purpose of the simulation study is only to determine the number of additional flow measurements required to identify the leaking node using our proposed procedure. The number of additional flow measurements depends on the location of the leaking node, and the average number of flow measurements over all possible locations of the leaking node is therefore computed. The summary statistics for the average number of flow measurements required for different networks are reported in Tables 2 and 3.

Table 2
Results of ILP using goal-programming method.

Network	Number of measurements				
	Mean	Median	Mode	Max	Std
Exnet	29.74	31	34	42	5.88
CO. Springs	23.78	22	22	38	4.92
Richmond	11.80	11	10	20	2.23
Dtown	11.10	11	10	16	1.60
Bangalore	10.44	10	10	13	1.10

Table 3
Results of approximation algorithm.

Network	Number of measurements				
	Mean	Median	Mode	Max	std
Exnet	54.58	53	50	71	6.40
CO. Springs	35.90	33	31	51	6.81
Richmond	13.56	13	13	23	3.38
Dtown	12.26	12	12	18	1.78
Bangalore	10.31	10	9	13	1.37

For the goal programming formulation, a nominal goal of $\gamma = 0.1$ was chosen. This corresponds to the requirement that both partitions have at least $0.4n$ nodes. As expected, the number of queries required are minimal when solving the goal programming ILP. It is also observed that the approximation algorithm performs reasonably well, with the number of queries much smaller than the size of network (both nodes and edges). As an example, when using the goal programming ILP formulation, for the largest network Exnet, the maximum number of queries required is only 1.7% of the number of edges. For the Bangalore network, which is the smallest considered, about 8.4% of the edges need to be queried in the worst case. This is expected since in small networks or sub-networks, modular features are less prominent. Similar trends are observed when using the approximation algorithm as well. Under worst case scenario, the fraction of queries (against edges) required when using the approximation method is 2.9% for Exnet and 8.4% for Bangalore.

5.2. Case study 2 – leak in edges/pipes

In most practical cases, it is unlikely that leaks can be present only in nodes. In general, we would not know upfront the nature of leak (whether it is at a node or on a pipe). For this case study, we again run the algorithm till we find the leaky pipe, though it can be stopped prematurely if required. As an illustration consider the situation shown in Fig. 12(a) which contains a leaky pipe shown in red. We query different edges in sequence as shown in Fig. 12(b)–(d) where the thick black lines indicate pipes that are queried. The

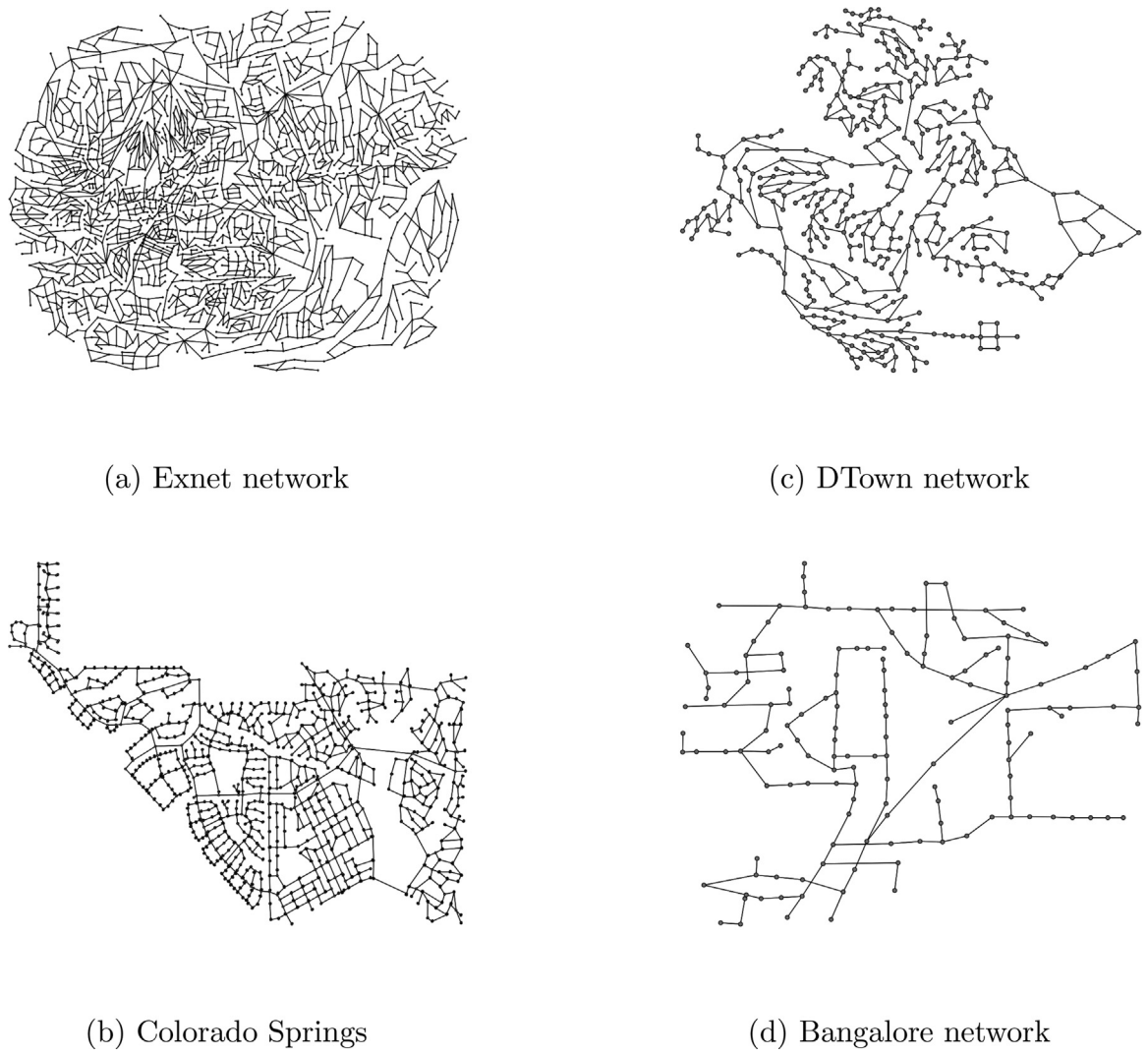


Fig. 11. Layouts of the different networks considered in this paper. Network structure and layout was obtained from the website of The Centre for Water Systems at the University of Exeter.

process is continued till we converge to the leaky pipe. We again perform a full enumerative study to generate the results presented in Tables 4 and 5.

For goal programming, we set $\gamma = 0.1$. Similar trends to Case Study 1 are observed where the ILP GP formulation performs well and requires only a small fraction of the pipes to be queried in order to identify the leaky pipe. For the largest network, only 2.3% of edges are queried, and for the smallest network, about 11% of the edges are queried, in the worst case when using the goal programming ILP formulation.

Table 4
Results of ILP using goal-programming method.

Network	Number of measurements				
	Mean	Median	Mode	Max	Std
Exnet	34.00	35	34	55	6.73
CO. Springs	26.20	25	24	45	5.11
Richmond	14.00	13	12	25	2.85
Dtown	13.13	13	12	22	2.31
Bangalore	12.10	12	12	17	1.28

Table 5
Results of approximation algorithm.

Network	Number of measurements				
	Mean	Median	Mode	Max	Std
Exnet	48.37	50	59	85	13.32
CO. Springs	39.52	38	31	59	7.93
Richmond	15.50	15	14	35	3.95
Dtown	15.35	15	14	29	3.18
Bangalore	12.00	12	11	18	1.65

6. Extensions to proposed approach

In this section we describe modifications to the basic approach we have proposed to extend its applicability to more general situations. We also describe some possible future extensions to improve the performance of our proposed approach.

6.1. Extension to multiple leak detection

In our proposed approach, we assumed a leak in only one node or pipe. The proposed algorithm can be naturally extended to cases when there are multiple leaks. In such a scenario, more than one sub-network would show an imbalance at some stage of the

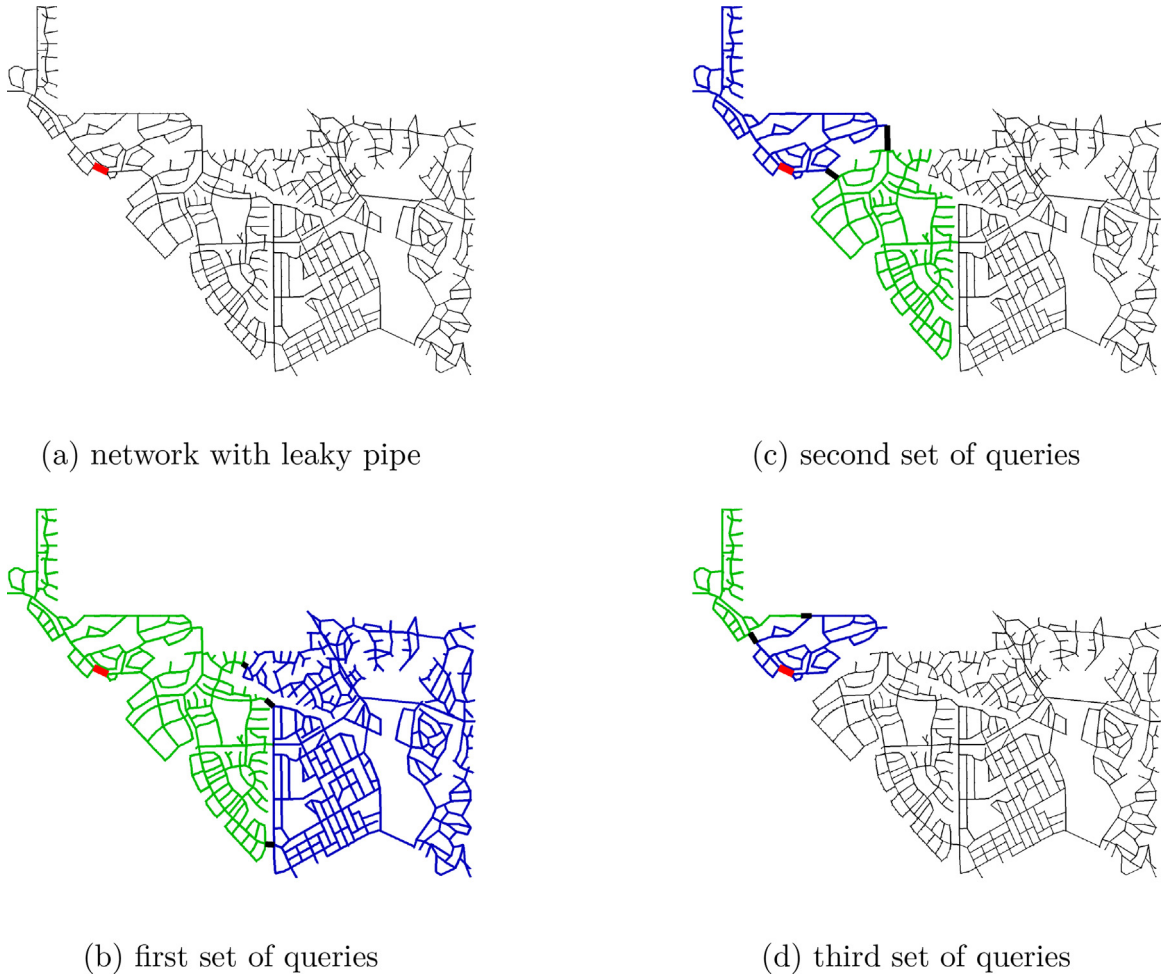


Fig. 12. Sequence of pipes that are queried to identify the leak shown in (a) in red. (For interpretation of the references to color in this legend, the reader is referred to the web version of the article.)

hierarchical partitioning exercise. After this stage, we apply the partitioning strategy to each of these sub-networks with imbalances to identify the leaking node within each sub-network.

6.2. Dealing with measurement noise

In our proposed approach we assumed that all flow measurements are error free. However, in practice all measurements will inevitably be corrupted by random noise. In this case, a flow imbalance around any partition cannot be presumed to be a result of a leak. A hypothesis test can be used (such as the nodal test proposed by Mah et al. (1976)) to decide whether the imbalance is greater than a selected threshold to conclude the existence of a leak within a partition, and accordingly probe the partition further. A knowledge of measurement error variances is required to choose the threshold using a statistical basis. Although, no change is required in the proposed sequential partitioning strategy, the method will give rise to both false alarms and missed detections. As part of future work, a Monte Carlo approach may be used to estimate the overall number of false alarms and missed detections resulting from the proposed approach.

6.3. Use of existing sensors

In the original problem formulation and algorithm, we assumed that pre-installed sensors are not available. Whenever a measurement was required, a query or act of measurement must be performed to obtain the flow rate. However, for well designed WDNs, some pipes would already be fitted with permanent flow sensors. This could be for DMA sectorization, or other monitoring requirements. In addition to sensors, we can also make use of valves by completely closing the valve through which we indirectly know that the flow rate in that pipe is zero. If such a disruptive method is not desirable, then the use of valves can be avoided. One method to incorporate the existence of these sensors, is to simply assign a very low querying cost to those pipes which have valves or sensors installed on them so that partitions containing them are favored over others. An extreme case of this is to simply remove those edges which have sensors on them from the network before running the partitioning algorithm and then use the appropriate flow rates when performing the water balance.

6.4. Different partitioning criteria

In our work, we have tried to obtain partitions that are balanced in size of the sub-networks (measured in number of nodes). There are possibly alternative criteria for balanced partitions that take into account domain specific knowledge. For instance, if a probability distribution for leak occurrences in various nodes are available, we might want to obtain partitions that are balanced in this probability. This information could be obtained for instance through historical data or models utilizing network properties like pipe lengths, roughness factors etc. For instance, total length of pipe in a partition could be related to the probability of leak occurrence

within the partition. It is easy to observe that node properties (like leak probability) can be easily incorporated into the ILP and approximate algorithms. However, it is not trivial to partition based on edge attributes (like pipe length) which is a line of work that can be pursued in future.

7. Conclusions

An effective graph partitioning based algorithm to identify leaking nodes or pipes in water distribution networks is proposed. The algorithm involves solving a multi-objective optimization problem that approximately models hierarchical graph partitioning. It was observed that a goal programming formulation handles the multiple objectives in an effective manner, producing high quality solutions. An approximate partitioning algorithm inspired by spectral clustering was also presented, and the results discussed. The performance of the algorithm and various formulations was elucidated through case studies on standard water distribution networks. It was observed that only a very small fraction of pipes need to be queried for flow measurements, in order to find the leak location.

Acknowledgements

This work was partially supported by the Department of Science and Technology, India under the Water Technology Initiative (DST/TM/WTI/2K13/144) and the IIT Madras Interdisciplinary laboratory for data sciences (CSE/14-15/831/RFTP/BRAV).

Appendix A. Review of algebraic graph theory

Before mathematically formalizing the problem, we briefly review relevant parts of algebraic graph theory. Specifically, we review the representation of WDNs as graphs and matrices, and survey useful properties. A more comprehensive description is available in the book by Deo (1974).

Definition 1. A graph, **G**, is a tuple **G**(**N**, **E**) comprising the set of vertices **N**, and edges **E** which are 2-element subsets of **N**. The number of vertices and edges in the graph are n and m respectively. The graph could be directed or undirected. We use the following terms interchangeably to suit the context: (i) graph and network; (ii) vertex, node, and junction; (iii) edge, link, and pipe.

The nodes of the network can be classified as source nodes where water is fed into the network, demand nodes or sink nodes where water is removed from the network for supplying to the consumers, and transmission nodes which aid in redistributing the flows. The edges of the network represent the pipes of the WDN. We choose an *undirected graph* representation for the network. However, we associate a sign convention with each edge to help identify the direction of flow. Flow will be negative if it is in the opposite direction to the chosen sign.

Definition 2. The adjacency matrix is defined by the relationship: $A_{ij} = 1$ if nodes i and j are connected by a pipe and 0 otherwise.

Definition 3. The directed incidence matrix **J** is defined by the relationship:

$$J_{ik} = \begin{cases} +1 & \text{if edge k connects nodes i and j, and } i < j \\ -1 & \text{if edge k connects nodes i and j, and } i > j \\ 0 & \text{if edge k is not incident on node i} \end{cases}$$

The sign convention for **J** can in fact be chosen arbitrarily and the above assignment is only one particular choice. The adjacency and incidence matrices characterize the network completely.

Definition 4. The degree of node i, is the number of edges incident on the node and denoted by deg(i). The degree matrix **D** is a diagonal matrix containing the degree of each node along the diagonal entries, i.e., $D_{ii} = deg(i)$ and $D_{ij} = 0$, $i \neq j$.

Definition 5. The Laplacian (**L**) of a graph is defined by the relationship $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where **D** and **A** are the degree and adjacency matrices, respectively.

Definition 6. A subgraph $\mathbf{S(N_S, E_S)}$ is formed from a graph **G(N, E)** such that $\mathbf{E_S} \subseteq \mathbf{E}$, and $\mathbf{N_S} \subseteq \mathbf{N}$ contains all the nodes on which the edges of $\mathbf{E_S}$ are incident.

Definition 7. A partition of **G(N, E)** consists of two subgraphs **S** and $\bar{\mathbf{S}}$ such that $\mathrm{N}_{\bar{\mathrm{S}}} = \mathbf{N} \backslash \mathrm{N}_{\mathrm{S}}$

Definition 8. The cut-set of partition $(\mathbf{S}, \bar{\mathrm{S}})$ is the set of all edges having one of their incident nodes in $\mathbf{N_S}$ and the other in $\mathrm{N}_{\bar{\mathrm{S}}}$. Formally, $\mathrm{cut}(\mathbf{S}, \bar{\mathrm{S}}) = \mathbf{E} \backslash \left(\mathrm{E_S} \cup \mathrm{E}_{\bar{\mathrm{S}}}\right)$

Definition 9. If each edge is associated with a cost, then the cut-cost of partition $(\mathbf{S}, \bar{\mathrm{S}})$ is the sum of costs of each edge present in the cut-set. We denote this with $R(\mathbf{S}, \bar{\mathrm{S}})$

. With these definitions, we can now formally define the graph partitioning problem.

Definition 100. Graph partitioning problem: Find a partition that minimizes $R(\mathbf{S}, \bar{\mathrm{S}})$, while satisfying certain constraints, such as cardinality constraints on $\mathbf{N_S}$.

Definition 11. A partition is said to be balanced if the number of nodes in the partitions are approximately equal, i.e. $|\mathrm{N_S}| \approx |\mathrm{N}_{\bar{\mathrm{S}}}|$.

We also review useful properties of these quantities, which will be used subsequently while developing the algorithms.

Property 1. *The Laplacian matrix is positive semi-definite.*

Property 2. *The smallest eigenvalue of the Laplacian matrix is 0. The vector* $\mathbf{v} = [1, 1, \ldots, 1]^T$ *(or simply* $\mathbf{v} = \mathbf{1}$*) satisfies* $\mathbf{Lv} = \mathbf{0}$ *and hence is an eigenvector corresponding to the 0 eigenvalue and belongs to the nullspace of L.*

Property 3. *The number of times 0 appears as an eigenvalue of the Laplacian (both algebraic and geometric multiplicity) is the number of connected components in the graph. These properties are straightforward to infer from the following proposition, for which we provide a simple proof here. A more thorough treatment of algebraic graph theory can be obtained from texts dedicated to it, such as Deo (1974).*

Proposition. *The Laplacian matrix is identically equal to the positive semi-definite matrix* $\mathbf{JJ}^T$

Proof. Define $\mathbf{Z} = \mathbf{JJ}^T - \mathbf{L}$ with eigenvalues $\lambda_1, \ldots, \lambda_n$ and corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. From direct verification, $\mathbf{x}^T\mathbf{JJ}^T\mathbf{x} = \sum_{(i,j)\in\mathbf{E}}(x_i - x_j)^2 = \mathbf{x}^T\mathbf{Lx}\ \forall\,\mathbf{x}$. Hence, $\mathbf{x}^T\mathbf{Zx} = 0\ \forall\mathbf{x}$ and in particular holds true for the eigenvectors, i.e., $\mathrm{v_i^T Z v_i} = \lambda_i \mathrm{v_i^T v_i} = 0$. This implies that all eigenvalues λ_i are 0 and hence $\mathbf{Z} = \mathbf{0}$ implying $\mathbf{JJ}^T = \mathbf{L}$.

References

Arora, S., Rao, S., Vazirani, U., 2009. Expander flows, geometric embeddings and graph partitioning. J. ACM 56 (2), 51–537.
Bhushan, M., Narasimhan, S., Rengaswamy, R., 2008. Robust sensor network design for fault diagnosis. Comput. Chem. Eng. 32 (45), 1067–1084.
Bhushan, M., Rengaswamy, R., 2002. Comprehensive design of a sensor network for chemical plants based on various diagnosability and reliability criteria. 1. Framework. Ind. Eng. Chem. Res. 41 (7), 1826–1839.
Colombo, A., Karney, B., 2002. Energy and costs of leaky pipes: toward comprehensive picture? J. Water Resour. Plan. Manag. 128 (6), 441–450.
Colombo, A.F., Lee, P., Karney, B.W., 2009. A selective literature review of transient-based leak detection methods. J. Hydroenviron. Res. 2 (4), 212–227.
Deo, N., 1974. Graph Theory with Applications to Engineering and Computer Science. Prentice-Hall, Inc.
Garcia, D., You, F., 2016. The water-energy-food nexus and process systems engineering: a new focus. Comput. Chem. Eng. 91, 49–67.

Gonzalez-Gomez, F., Garca-Rubio, M.A., Guardiola, J., 2011. Why is non-revenue water so high in so many cities? Int. J. Water Resour. Dev. 27 (2), 345–360.
Guruswami, V., Sinop, A., 2011. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)., pp. 482–491.
Kernighan, B.W., Lin, S., 1970. An efficient heuristic procedure for partitioning graphs? Bell Syst. Tech. J. 49 (2), 291–307.
Kim, Y., Lee, S., Park, T., Lee, G., Suh, J.C., Lee, J.M., 2016. Robust leak detection and its localization using interval estimation for water distribution network. Comput. Chem. Eng. 92, 1–17.
Liggett, J., Chen, L., 1994. Inverse transient analysis in pipe networks? J. Hydraul. Eng. 120 (8), 934–955.
Mah, R., Stanley, G., Downing, D., 1976. Reconciliation and rectification of process flow and inventory data. Ind. Eng. Chem. Process Des. Dev. 15, 175–183.
Mann, A., McKenna, S., Hart, W., Laird, C., 2012. Real-time inversion in large-scale water networks using discrete measurements. Comput. Chem. Eng. 37, 143–151.
Mpesha, W., Gassman, S., Chaudhry, M., 2001. Leak detection in pipes by frequency response method? J. Hydraul. Eng. 127 (2), 134–147.
Mulholland, M., Latifi, M., Brouckaert, C., Buckely, C., 2014. Leak identification in a water distribution network using sparse flow measurements. Comput. Chem. Eng. 66, 252–258.
Palleti, V., Narasimhan, S., Rengaswamy, R., Teja, R., Bhallamudi, S., 2016. Sensor network design for contaminant detection and identification in water distribution networks. Comput. Chem. Eng. 87, 246–256.
Pothen, A., Simon, H., Liou, K., 1990. Partitioning sparse matrices with eigenvectors of graphs? SIAM J. Matrix Anal. Appl. 11 (3), 430–452.
Puust, R., Kapelan, Z., Savic, D.A., Koppel, T., 2010. A review of methods for leakage management in pipe networks? Urban Water J. 7 (1), 25–45.
Rojas-Torres, M., Napoles-Rivera, F., Ponce-Ortega, J., Serna-Gonzalez, M., El-Halwagi, M., 2014. Optimal design of sustainable water systems for cities involving future projections. Comput. Chem. Eng. 69, 1–15.
Sherali, H., Soyster, A., 1983. Preemptive and nonpreemptive multi-objective programming: relationship and counterexamples? J. Optim. Theory Appl. 39 (2), 173–186.
Shi, H., You, F., 2016. Energy optimization of water supply systems scheduling: novel MINLP model and efficient global optimization algorithm? AIChE J. 62 (2), 4277–4296.
Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intel. 22 (8), 888–905.
Stephens, M., Lambert, M., Simpson, A., Vitkovsky, J., Nixon, J., 2004. Field Tests for Leakage, Air Pocket, and Discrete Blockage Detection Using Inverse Transient Analysis in Water Distribution Pipes., pp. 1–10 (Chapter 471).
Stephens, M., Simpson, A., Lambert, M., Vtkovsk, J., 2005. Field Measurements of Unsteady Friction Effects in a Trunk Transmission Pipeline., pp. 1–12 (Chapter 18).
Sun, Z., Wang, P., Vuran, M.C., Al-Rodhaan, M.A., Al-Dhelaan, A.M., Akyildiz, I.F., 2011. Mise-pipe: magnetic induction-based wireless sensor networks for underground pipeline monitoring. Ad Hoc Netw. 9 (3), 218–227.
Yazdani, A., Jeffrey, P., 2011. Complex network analysis of water distribution systems? Chaos: Interdiscip. J. Nonlinear Sci. 21 (016111), 1–9.