IISE Transactions

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie21

A scalable Bayesian framework for large-scale sensor-driven network anomaly detection

Feiran Xu & Ramin Moghaddass

To cite this article: Feiran Xu & Ramin Moghaddass (2022): A scalable Bayesian framework for large-scale sensor-driven network anomaly detection, IISE Transactions, DOI: 10.1080/24725854.2022.2037792

To link to this article: https://doi.org/10.1080/24725854.2022.2037792

IISE TRANSACTIONS
https://doi.org/10.1080/24725854.2022.2037792

A scalable Bayesian framework for large-scale sensor-driven network anomaly detection

Feiran Xu[a] and Ramin Moghaddass[a,b]

[a]Industrial and Systems Engineering, University of Miami, Coral Gables, USA; [b]Management Science, University of Miami, Coral Gables, USA

ABSTRACT

Many real systems have a network/graph structure with many connected nodes and many edges representing deterministic or stochastic dependencies and interactions between nodes. Various types of known or unknown anomalies and disturbances may occur across these networks over time. Developing real-time anomaly detection and isolation frameworks is crucial to enable network operators to make more informed and timely decisions and take appropriate maintenance and operations actions. To monitor the health of modern networks in real time, different types of sensors and smart devices are installed across these networks that can track real-time data from a particular node or a section of a network. In this article, we introduce an innovative inference method to calculate the most probable explanation of a set of hidden nodes in heterogeneous attributed networks with a directed acyclic graph structure represented by a Bayesian network, given the values of a set of binary data observed from available sensors, which may be located only at a subset of nodes. The innovative use of Bayesian networks to incorporate parallelization and vectorization makes the proposed framework applicable for large-scale graph structures. The efficiency of the model is shown through a comprehensive set of numerical experiments.

ARTICLE HISTORY
Received 17 June 2021
Accepted 16 January 2022

KEYWORDS
Anomaly detection; Bayesian networks; sensor analytics; system monitoring

1. Introduction

Many complex mechanical, electrical, social, communication, and computer systems are heterogeneous attributed networks, which are systems with a graph structure characterized by a set of nodes representing different types of objects, a set of edges representing relationships and dependencies (deterministic or stochastic) between objects, and a set of node attributes or features that may vary depending on the type of objects at each node. Many different types of smart sensors and remote sensing devices exist in such networks, to improve the ability to track the state of the network and help make real-time decisions. Since node attributes are often collected through sensors at many or all nodes of the networks, such networks are often sensor-intensive. Also, since the attributes collected at each node can differ, attributed networks can be heterogeneous in this regard. Various types of known or unknown anomalies and disturbances may occur across these networks over time that can impact the operating status of each node. Such an anomaly status can be defined through a binary setting, that is, a node is either anomalous or normal (not anomalous) at any point of time. The definition of anomaly varies depending on the application, but is often defined as patterns/events that do not conform to a well-defined notion of normal behavior. Intrusions and bad connections in wireless ad-hoc networks (Murugan and Suresh, 2017), water contamination in water distribution networks (Tuptuk *et al.*, 2021), outages and power disturbances in power distribution networks (Yuan *et al.*, 2020), and false product reviewers in social networks (Yu *et al.*, 2016), are all real examples of anomalies that can impact the nodes of their corresponding networks. Anomalies, which are alternately called outliers, disturbances, and abnormal behaviors, can be momentary or sustained. The process of network anomaly detection is often complex, as networks include dependent nodes and sensors that are influenced by the topological dependencies between nodes, edges, and sensor data. Despite the growing importance of sensor-based network anomaly detection, this area has received relatively little attention for heterogeneous attributed networks. Many anomaly detection models developed for these networks are general-purpose methods and are not designed to deal with the complexities of network structure and sensor data. Using them without considering network complexities can lead to too many false alarms or too few detected anomalies.

This work is highly motivated by the growing opportunities for real-time monitoring and control arising from the availability of sensor data across a broad range of network systems, particularly for power distribution networks, which are sensor-intensive networks that are subject to many types of anomalies and disturbances over time. Power distribution networks are large-scale graph structures with many sensors that are sensitive to various types of anomalies and disturbances, which can cause momentary or sustained outages. Quickly and accurately finding the set of anomalous nodes

CONTACT Ramin Moghaddass ramin@miami.edu

and the sources of anomalies is a critical task, which has significant impact on maintaining grid reliability and resiliency to avoid costly disruptions in the operations of critical infrastructures. How to analyze and merge network sensor data collected from sensors across the network and locate anomalies while considering the topological dependencies between nodes without the need for physical inspections is an important research and practical problem that needs further study. Our work is also motivated by some of the unique aspects of Bayesian Networks (BNs) that can be utilized to model simultaneously the topological structure of the heterogeneous attributed networks, the stochastic relationship between network status and sensor data, and the problem of anomaly detection in an interpretable and scalable manner.

In this article, a novel BN-based approach is proposed to diagnose a large network with many nodes and many sensors to locate the potential sources of anomalies and impacted nodes. A BN is a probabilistic graphical model (comprised of nodes and directed edges) that can represent the dependencies between a set of hidden or observable random variables with a directed acyclic graph (a directed graph with no cycle). The status of each node, the location of anomalies, and the values of sensors are represented as nodes of a BN and then the topological dependencies between nodes, the propagation of anomalies, and the stochastic behavior of sensor data are employed to form the directed edges that represent the conditional dependencies between variables in a directed graph. The proposed approach formulates the problem of network anomaly detection in the context of a BN and then develops computational techniques to make the approach scalable for large-scale networks, such as power distribution networks. The main contributions of our work can be summarized as follows:

1. We formulated an anomaly detection framework for large-scale network/graph structures with tree or non-tree topologies using BNs.
2. We developed an efficient Bayesian sampling procedure to calculate the posterior probabilities of a set of hidden nodes given a set of evidence nodes.
3. We developed vectorization and parallelization techniques to speed up inference propagation in large-scale and complex networks. Also, we proposed an algorithm to divide the network into multiple segments/clusters where the sampling for all nodes within each cluster can be conducted simultaneously.

In this work, we have focused only on binary sensor attributes collected at the node level, due to their wide application in many network structures, such as power distribution networks, water distribution networks, and communication networks. Also, our work considers only anomalies that originate from a set of nodes and then propagate to connected nodes across the network according to a known set of propagation paths. Thus, analyzing edge attributes and anomalies with random shapes not following propagation paths are beyond the scope of this article.

The rest of this article is organized as follows. In Section 2, we provide a review of the current literature and highlight the challenges of existing works. Section 3 discusses the problem of anomaly detection and the general structure of the proposed network anomaly detection framework. Section 4 introduces steps to scale up the proposed framework using parallelization and vectorization. In Section 5, we show the performance of our framework through a comprehensive set of numerical experiments. We conclude in Section 6 and discuss our future work.

2. Literature review

2.1. Anomaly detection in networks and graph structures

Anomaly detection has been widely studied by many researchers from diverse areas, and there are plenty of review and survey articles available for the work in this domain (see, for instance, Chandola *et al.* (2009)). Anomaly detection has been studied extensively in many application domains, including network intrusion detection (Bhuyan *et al.*, 2013), credit card fraud detection (Popat and Chaudhary, 2018), road traffic anomaly detection (Kim and Cho, 2018), manufacturing and production system anomaly detection (Stojanovic *et al.*, 2016), medical and public healthcare anomaly detection (Antonelli *et al.*, 2013), and sensor network anomaly detection (Janakiram *et al.*, 2006). Most available anomaly detection studies have focused on general-purpose anomaly or outlier detection methods and are not particularly designed for structures, such as networks or graphs. For example, anomaly detection can be simply formulated as a binary classification problem with the unrealistic assumptions that network data are independent and identically distributed and the same types of sensor data are collected from all network nodes. Due to the special structure of network/graph structures and the complexity of network sensor data, more research has recently been focused on developing models that are designed based on the specific features of networks.

In Akoglu *et al.* (2015) and Ranshous *et al.* (2015), an extensive survey of graph-based anomaly detection techniques in static and dynamic graphs and several real applications of graph-based anomaly detection methods (supervised and unsupervised) are provided. It is known that many of the network-based anomaly detection models have a relatively high false positive rate (Akoglu *et al.*, 2015), which makes the application of these methods on real systems more challenging. Available methods of anomaly detection can be categorized into the following classes: statistical detection models, such as histograms and control charts; classification-based models; clustering-based methods; and data mining and machine learning techniques. Bhuyan *et al.* (2013) presented an overview of techniques used in network anomaly detection models. Noble and Cook (2003) proposed two methods of detecting anomalies in graph-based data according to the repetitive patterns or substructures of the network. Janakiram *et al.* (2006) suggested an anomaly detection algorithm based on a Bayesian belief network by considering the conditional dependencies in wireless sensor networks. Hooi *et al.* (2019) proposed an online anomaly

detection algorithm given sensor data in the power grid system. All of these methods have unique strengths and weaknesses, and their performance varies depending on the application. For example, strong distributional assumptions in statistical models; ineffectiveness in distance measures in clustering and nearest neighbor methods, particularly in high-dimensional settings; high vulnerability of unknown anomalies in supervised classification models; and high false alarm rates and difficulty in model training in unsupervised models make these methods ineffective for complex graph structures. Another limitation of these approaches is that anomalies are treated generally, and their cause and type are unidentifiable.

2.2. Binary network sensor data

Binary sensors, which are noisy sensors that generate binary outcomes, are widely used due to their simplicity, efficiency, and relatively low cost, particularly in power distribution networks (Alghuried and Moghaddass, 2020), wireless sensor networks (Guerriero *et al.,* 2009), and water distribution networks (Ostfeld, 2008). In general, binary sensors may detect the presence or absence of a particular target in their sensing regions. The accuracy of the detection depends on the applications and the detection power of the sensors (Asadzadeh *et al.,* 2011). Examples of such binary signals are various types of SCADA messages (e.g., switches open/close), smart meter binary errors, and binary messages generated across power distribution networks. For example, before an actual outage or severe anomaly occurs, there are often events, such as transient faults, errors, voltage irregularities, or power quality events/errors, that are reported as binary signals and are often available for most residential and commercial smart meters. Many other power quality measurements are binary and are commonly reported in typical smart meters. For instance, the last gasp error is generated when a smart meter detects a zero voltage event lasting for a short period of time, and a voltage sag error is reported when there is a short drop of a root mean square voltage below a threshold for a period of time. In water distribution networks, noisy binary sensors with a fixed false positive rate and true positive rate are used to locate contaminant plumes (Speakman *et al.,* 2015). With the increasing use of smart and binary sensors in networks, research over the past decade has focused more on using binary sensor data for decision making. Wang *et al.* (2010) introduced a real-time distributed target tracking algorithm with binary sensor networks. Chen *et al.* (2011) proposed a practical algorithm for target tracking with binary detection sensors. Djuric *et al.* (2008) presented algorithms using particle filtering methods to track a single target in a binary sensor network. Zhang *et al.* (2020) constructed a Kalman filtering approach for an estimation problem for a binary sensor network. Despite the huge investment in deploying binary sensors in many large-scale networks, relatively less network-specific work has been conducted on developing analytical methods to utilize binary sensor data for anomaly detection.

2.3. BN Models for network anomaly detection

Due to the structure of the anomaly detection problem in attributed graphs and the fact that known and unknown variables in such graphs follow a highly interpretable and causal structure, BN inference is a great fit to model this problem in a structured manner. Considering sensor data as observations represented by observable nodes and nodes' anomaly status as explanation nodes that are hidden, BNs can be employed to formulate the anomaly detection problem in network structures. Finding explanations for a set of observations or evidence in Bayesian belief networks is a challenging problem that has attracted a lot of attention in the scientific community. A comprehensive introduction and overview of explanation methods for BNs, including the explanation of evidence, can be found in Lacave and Díez (2002). The explanation of evidence can be defined as the configuration of unobservable variables that can interpret the existing set of observations (Pearl, 1988). The process of finding such explanations is sometimes referred to as the abduction inference (Neapolitan 2004). A very popular approach for the inference problem in BNs is the Most Probable Explanation (MPE) method, which aims to find the most likely value of all possible combinations of variables given the subset of evidence. Kwisthout (2011) provided a comprehensive overview of the complexity and tractability of computing the MPEs in BNs. Neapolitan (2004) introduced a best-first search algorithm for abductive inference to find the MPEs using the branch-and-bound pruning technique in a state-space tree. Marinescu and Dechter (2012) demonstrated an extension of the best-first AND/OR graph search method from branch-and-bound in which the OR node represents variables, whereas the AND node represents the status of each variable. Yuan and Lu (2007) proposed a new approach to find the explanatory MAP (eMAP), which can identify the most relevant explaining variables by ranking the Bayes factor of each configuration for given observations in the BN. However, this method is not scalable and has difficulty handling networks with just hundreds of nodes. Li and D'Ambrosio (1993) presented a non-search approach for finding the MPEs in an arbitrary Bayesian belief network. An algorithm for finding the MPEs using the massage passing method in both singly connected and multiply connected networks was presented in Sy (1992). Mengshoel *et al.* (2010) discussed a local search approach to focus on initialization and restart to solve the MPE problem in BNs. It has been shown that the difficulty in finding the MPEs in BNs is NP-hard (Cooper, 1990). Almost all available MPE methods are not scalable for large-scale networks and can only be used in small network settings. Thus, using available heuristic and MPE-based methods for large-scale network anomaly detection models is not a feasible option. A new generation of scalable techniques is needed that can incorporate the complexity of today's network sensor data, the topological dependencies between nodes and sensor data, and the propagation properties of anomalies across the network. We will show that flexible BNs with many stochastic and deterministic nodes and edges are highly

Table 1. List of notation used in this article.

Notations	Descriptions
N	The set of nodes in the network
V	The set of edges in the network
$\mathbf{A} = [a_{ij}]_{\|N\|\times\|N\|}$	The adjacency matrix (representing physical/virtual connections between nodes)
J	The set of sensor attributes
S	The number of sensors in the network
$\mathbf{Z} = [z_{ij}]_{\|N\|\times J}$	The sensor-attribute matrix
z_{ij}	The binary index representing whether node i generates sensor output j
$y_{ij}(t)$	The value of sensor output j for node i at time t
$x_i(t)$	The binary index representing the anomaly status of node i at time t
$o_i(t)$	The binary index representing whether node i is the source of anomaly at time t
$\mathbf{P} = [p_{ij}]_{\|N\|\times\|N\|}$	The anomaly propagation matrix
p_{ij}	A binary index representing whether an anomaly propagates from node i to node j
D_n	The set of downstream/descendant nodes of node n
D_n^c	The set of immediate children nodes of node n
U_n	The set of upstream/ancestor nodes of node n
U_n^p	The set of immediate parent nodes of node n

interpretable and efficient tools for modeling the problem of anomaly detection.

2.4. Summary of available methods and their limitations

Many efforts have been made in the literature on anomaly detection for attributed networks, each with positive properties and serious limitations. Since network systems and graphs and network sensor data are typically complex, due to various reasons (e.g., large-scale and heterogeneous sensor data and topological dependencies between network elements), available general-purpose methods are not effective for anomaly detection in attributed networks, and thus, network-specific models that can adapt to such complexities must be developed. Among those methods that are developed for network systems, there are three major categories: network classification, node classification, and subgraph classification. Although both network classification and node classification problems are useful and have important applications, they are limited in the sense that they cannot find the sections (subgraphs) where the anomalous nodes are located. The methods that are designed for subgraph anomaly detections also have at least one of the following common limitations: not simultaneously using sensor and graph topology data, not applicable to all network structures or network data types, not scalable for large graphs, ignorance of anomalous nodes' connectivity and anomaly propagation, and inability to detect the source(s) of potential anomalies. In addition, since many of the models in this category just rank nodes and subgraphs based on a predefined anomaly score, they cannot be used to distinguish healthy networks from anomalous networks. Also, some of these methods require specific parameters as inputs, such as the size and shape of the detected subgraphs, which are hard to determine in practice. In Table 1 in the Supplemental Materials, we summarize the main characteristics of a series of recent and relevant works on anomaly detection for static attributed networks and reported their objective/output, type (supervised, unsupervised, and semi-supervised), main approach, data used, and main drawbacks and application limits. It is clear from this table that each method has some limitations that narrow its potential use for network anomaly detection. The proposed framework introduces new interpretable ways to detect anomalies in networks and graph structures while addressing the limitations in the available literature. Although a lot of research has been conducted on network anomaly detection for attributed networks, efficient use of large BNs for anomaly detection in attributed and heterogeneous networks is novel, particularly for large networks with topological dependencies.

3. Main approach

This section illustrates the details of the proposed anomaly detection framework, which is characterized by a Bayesian network. The list of notation used in this article is provided in Table 1.

3.1. Problem definition and assumptions

In this subsection, we list the main assumptions made in this article to better define the scope and the application of this article. The system under study has a graph/network structure made up of nodes or vertices connected by links or edges. The network topology can be represented by a known Directed Acyclic Graph (DAG), which is a directed graph with no directed cycles and tree or non-tree topology. An anomaly is defined as an abnormal behavior or a deviation from normal behavior that can impact the binary operating status of each node. In other words, at each point of time, a node can either be normal or anomalous. All anomalies are assumed to originate at one or a group of nodes and then propagate to connected nodes across the network and make them anomalous according to a known set of propagation paths (Section 3.2.2). A set of noisy binary sensor attributes (if any) may be collected at each node (edge attributes are not considered). Each sensor attribute may only partially reveal the anomaly state of the node where the sensor is located. It is assumed that an anomaly cannot be detected directly from available sensors. Since the sensor attributes available at each node are not the same across all network nodes, the network under study falls under the category of a heterogeneous attributed network.

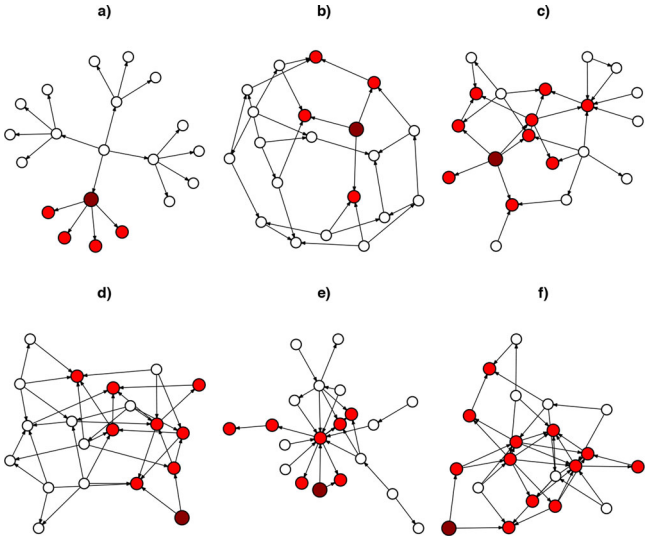

Figure 1. Examples of anomaly propagation in randomly generated tree (a) and non-tree (b-f) structures. The dark red nodes are the origins of anomalies and the red nodes are anomalous nodes.

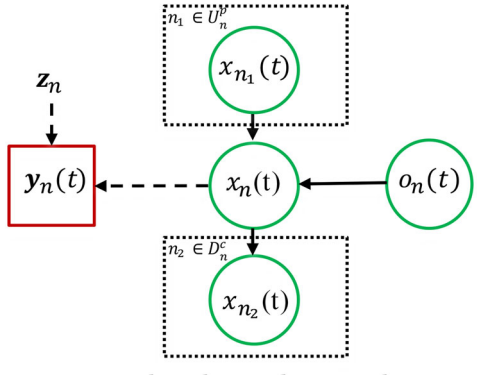

Figure 2. A graphical model representing the dependencies between BN variables for node n and its immediate children (D_n^c) and parents (U_n^p). Circles are hidden nodes, and rectangles are sensor nodes. Solid lines represent deterministic relationships, and dashed lines represent stochastic relationships. Only the connection from and to node n and its corresponding variables are shown.

The topology of the network is known and remains unchanged over time. Below, we define the generic problem of anomaly detection for an attributed network, which is a graph structure with sensor outputs/attributes at the nodes of the network. Then, we discuss the main elements of the proposed framework.

Problem 1. Anomaly Detection for Heterogeneous Networks. Given the topology of an attributed network, sensor locations and attributes, and anomaly propagation sets for all network nodes, aggregate/analyze real-time network sensor observations collected from sensors across the entire network to identify (i) the most likely sources of potential anomalies (if any), (ii) the impacted anomalous nodes/subgraphs, and (iii) the uncertainty associated with the results from (i) and (ii).

3.2. A generic structure for attributed networks

The characteristic variables/parameters and the assumptions made to characterize a network, its topology, and the detection framework are discussed in this subsection. A heterogeneous attributed network or graph G, represented by $G = (\mathbf{A}, \mathbf{P}, \mathbf{Z}, \mathbf{Y})$, is a system made up of nodes or vertices connected by links or edges that are characterized by the adjacency matrix $\mathbf{A}$, the anomaly propagation matrix $\mathbf{P}$, the sensor-attribute matrix $\mathbf{Z}$, and the sensor data matrix $\mathbf{Y}$.

3.2.1. Graph topology

A network, or graph G, is a system made up of a set of nodes or vertices, denoted by N, connected (virtually or physically) by a set of links or edges, denoted by V. Here, V can be obtained by an $|N| \times |N|$ adjacency matrix $\mathbf{A} = [a_{ij}]$, where $a_{ij} = 1$ if there is a direct link (edge/arc) from node i to node j, that is, if $\{i,j\} \in V$. Each node corresponds to a unique entity, such as a sensor, device, equipment, or user, and can be a potential candidate for the original location of an anomaly. The topological structure of the network should follow a DAG which is a directed graph with no cycles.

3.2.2. Anomaly and its propagation

At any point of time, anomalies can originate from one or multiple nodes of the network. Thus, at each point of time, a node may be anomalous or not anomalous. Due to the topological structure of the network and dependencies between nodes, anomalies can propagate across the network. The propagation of anomalies in the network is characterized by a set of anomaly propagation paths/sets, which can be represented by a $|N| \times |N|$ matrix $\mathbf{P} = [p_{ij}]$, where $p_{ij} = 1$ if an anomaly can propagate from node i to node j. In a deterministic case, when node n experiences an anomaly, the linked neighbor nodes may or may not experience the same anomaly, that is, p_{ij} is a binary variable being one when an anomaly propagates from node i to node j. For example, in a distribution network, the propagation of any outage in the network is fully known according to the location of protective devices. Similarly, the propagation of the contamination in a water distribution network depends on the topology of the network and the points (sources) of contamination. In other words, contamination at a point in the network affects downstream locations in the network. Due to the sparse structure of most large-scale networks, only positive p_{ij}s may be recorded as a list variable. In the case of stochastic propagation, p_{ij} can represent the probability that an anomaly propagates from node i to node j. It is important to note that for any node in the network, matrix $\mathbf{P}$ considers all impacted nodes and not just its neighbors. We assume that $\mathbf{P}$ is either known or can be estimated from past data. In the stochastic case, which is beyond the scope of this article, we have a probabilistic graphical model where the conditional dependencies can be shown via a DAG. In our framework, downstream/descendant nodes, upstream/ancestor nodes, parent nodes, and children nodes are defined according to the anomaly propagation paths. For instance, node m is a child of node n if $a_{nm} = 1$ according to the adjacency matrix and any anomaly at node n propagates to node m (that is $p_{nm} = 1$). In such a case, n is the parent of node m. Since for any node, we can have multiple

parents or children, we use U_n and D_n to denote the set of ancestors and descendants of node n, respectively. Based on this notation set, node m is a direct child of node n if $m \in D_n^c$ and $a_{nm} = 1$. Also, if node $k \in D_n^c$, then $p_{nk} = 1$ and if node $k \in U_n$, then $p_{kn} = 1$. We should point out that anomaly propagation can occur in both tree and non-tree structures. To better illustrate this point, six examples of anomaly propagation for random tree and non-tree structures are shown in Figure 1.

3.2.3. Network monitoring variables

To monitor a network structure with regards to anomalies, a health state variable is defined for all nodes based on the topology of the network. For node n, variable $x_n(t)$ is defined to indicate whether node n is under the effect of an anomaly at time t. To monitor the original locations of anomalies, we also define a binary variable $o_n(t)$ reflecting whether node n is the original location/source of the anomaly in epoch t. It is clear that if $o_1(t), o_2(t), ..., o_{|N|}(t)$ are known, the anomaly status of all nodes, represented by $x_1(t), x_2(t), ..., x_{|N|}(t)$, are known as well (based on the deterministic anomaly propagation paths). These network status variables are not directly observable and only sensor data may be available to partially infer their status.

3.2.4. Graph sensor data

The data collected across a network are of various types and are generated from various sensors. Sensors are located at a known subset of nodes. There are a total of $|J|$ types of binary sensor attributes (outputs) collected in the network, which may be generated from $|J|$ or less than $|J|$ types of sensors (i.e., each sensor may generate one or more types of binary sensor attributes). There may be multiple sensors located at any given node, however, we mainly track the types of sensor attributes available at each node. This work focuses only on binary attributes due to their wide application in many network structures, such as power distribution and communication networks. We refer to the network structure as an attributed heterogeneous network because attributes are available at subsets of nodes and the sensor attributes at any two nodes may be different. It is assumed in this article that a binary attribute can either be (i) the direct outcome of a sensor (e.g., one if there is electric current or zero if there is no electric current in a smart meter); or (ii) a transformed representation of a non-binary sensor attribute (or attributes) that is found by either preprocessing original sensor values (e.g., applying a threshold to classify temperature sensor outcomes to high and low) or feature transformation (e.g., applying one-hot encoding on a categorical feature to transform it into a binary feature vector). Networks with non-binary sensor data that are not convertible to binary data are beyond the scope of this work. During any time interval of interest, node n is assumed to generate either the output $y_{nj}(t)$ or simply no output (if no sensor that generates output j is at this node or if there is a missing point), that is, $y_{nj}(t) = NA$, for sensor output j. The binary index z_{nj} represents whether node n generates feature j or whether the jth sensor attribute is collected at node n. With this notation, users can set z_{nj} to zero when sensor attribute j is not collected at node n (when the sensor that generates attribute j is not installed at node n) or when that specific data point is missing. Since the status of each node is binary, the outcome of each sensor attribute is binary, and sensor attributes are assumed to be conditionally independent given the node's binary status. Further, the relationship between sensor attributes and node status can be mathematically defined with two parameters, the false positive rate (denoted by β_j), and the true positive rate (denoted by α_j), as follows:

$$\begin{aligned}
&\Pr(y_{nj}(t) = 1 | x_n(t) = 1, z_{nj}(t)) = z_{nj}(t)\alpha_j, \\
&\Pr(y_{nj}(t) = 1 | x_n(t) = 0, z_{nj}(t)) = z_{nj}(t)\beta_j, \\
&\forall n \in N, \quad \forall j \in J.
\end{aligned}$$

Modeling binary sensor behavior using the above equations is very reasonable and has been extensively used in the literature. The power of these sensor attributes is defined through parameters α and β. Based on this sensor behavior model, the larger α and $(1-\beta)$ are, the stronger are the corresponding sensors in distinguishing anomalies from normal instances. Now, based on the conditional independence assumption of sensor data at each node given the status of that node, we can build the following sensor model (assuming that $z_{n1}(t) = 1, ..., z_{n|J|}(t) = 1$):

$$\begin{aligned}
&\Pr(y_{n1}(t) = v_1, ..., y_{n|J|}(t) = v_{|J|} | x_n(t) = x) \\
&\quad = \left[\prod_{j \in J} \alpha_j^{v_j}(1-\alpha_j)^{1-v_j}\right]^x \left[\prod_{j \in J} \beta_j^{v_j}(1-\beta_j)^{1-v_j}\right]^{1-x}, \\
&\forall x \in \{0, 1\}.
\end{aligned}$$

3.3. Problem statement in the context of BNs

In this subsection, we discuss how the problem of anomaly detection for an attributed heterogeneous network can be framed through a BN with both deterministic and stochastic edges as well as hidden and observable nodes. A BN is a probabilistic graphical model (comprised of nodes and directed edges) that can represent the dependencies between a set of hidden or observable random variables with a DAG (a directed graph with no cycle). The proposed BN modeling will enable network operators to obtain integrated insight regarding the entire network and the uncertainty associated with that insight at any decision epoch. Given a heterogeneous attributed network with a known topology and sensors, locations, and attributes, the objective is to analyze real-time network sensor data, from all or a subset of sensors across the network to find the most likely status of the network (with regards to anomalies). The full status of the network is defined based on the status of all nodes, potential anomalies, and the original sources of the potential anomalies. This problem can be extended to finding the K most likely scenarios with regards to the status of the entire network given available sensor data.

3.3.1. BN and its connection with the network topology

The causal relationship between the network topology elements, anomaly and its propagation, monitoring variables, and sensor data can be represented by a BN through a DAG as shown in Figure 2. The directions of edges represent the dependency/causal relationships, where the initial nodes are the causes of destination nodes. It is clear from Figure 2 that each node may have multiple parents and multiple children as discussed in Section 3.2.2, thus, the graph structure is not limited to tree structures unlike many available graph anomaly detection frameworks that work only on simple graph structures, such as trees, with both deterministic and stochastic edges, as well as hidden and observable nodes. There are three stochastic variables in the BN for each node n of the original graph (that is o_n, x_n, and $\boldsymbol{y}_n$). Each variable in the BN is represented by a distinct node. There are two types of variables/nodes in this BN. The rectangles represent observable/evidence nodes (i.e., sensor nodes $\boldsymbol{y}$), and the circles represent the hidden nodes (i.e., network monitoring variables x and o). In BNs a node whose status has been observed with probability 1 is an evidence node. It is important to note that all sensor outputs at each node are represented in one box ($\boldsymbol{y}$), which reflects the fact that they are not necessarily independent. There are two types of edges in the network as well, namely, stochastic edges (dashed lines) and deterministic edges (solid lines). The deterministic edges are defined based on the network topology and anomaly propagation paths. For instance, the solid edge between $o_n(t)$ and $x_n(t)$ ensures that if $o_n(t) = 1$ then $x_n(t) = 1$. Also, $x_n(t) = 1$ if any of its ancestors (U_n^p) are impacted by the anomaly. The stochastic edges between x and y represent the relationship between the status of each node and sensor outputs for that node (if there is any). Also, the binary parameter z_{nj} in $\boldsymbol{z}$ indicates whether node n generates feature j or whether the jth sensor attribute is collected at node n. In Figure 3, the BN transformation of a small section of a distribution network is presented. It is clear that the topology of the original graph/network and the topology of the BN are not the same, because the BN combines network variables with topology parameters and their relationships, and thus has more nodes (three times more nodes) and edges (much more depending on the structure of the anomaly propagation in the network). It can be seen from this figure that a physical network can be modeled by a casual graph representing the propagation of anomalies (middle graph) and a BN. For each node in the original network, there are three variable nodes in the BN.

Each variable/node in a BN is conditionally independent of all its non-descendants in the graph given the value of all its parents. To fully specify the BN, it is necessary to characterize each edge that is specifying the probability distribution or causal relationship between each node conditional upon its parents. In this article, instead of using a conditional probability table to show the relationship between Boolean variables $\boldsymbol{x}$ and $\boldsymbol{o}$, we define a set of mathematical constraints to characterize the deterministic edges. The set of constraints that guarantees the causal dependencies (or conditional independence relationships implied by the graph) between elements $\boldsymbol{x}$ and $\boldsymbol{o}$ can be mathematically defined with a set of binary constraints for graph G as follows:

$$\Phi(G,\boldsymbol{x},\boldsymbol{o}):\begin{cases} o_n + \sum_{m\in U_n^p} x_m \le (|U_n^p|+1)x_n & \forall\ n\in N; \\ x_n \le o_n + \sum_{m\in U_n^p} x_m & \forall\ n\in N. \end{cases} \tag{1}$$

The first constraint implies that if node n is the original location of an anomaly (i.e., $o_n = 1$) and/or at least one of the direct parents of node n is anomalous (i.e., $\sum_{m\in U_n^p} x_m \ge 1$), then node n is impacted by an anomaly (i.e., $x_n = 1$). The second constraint ensures that if node n is not the source of an anomaly (i.e., $o_n = 0$) and all its direct parents are healthy (i.e., $\sum_{m\in U_n^p} x_m = 0$), then node n is in a healthy condition (i.e., $x_n = 0$). From a network point of view, the first constraint implies that an anomaly propagates from a node to all of that node's descendants, and the second constraint implies that any node can be impacted by an anomaly originating at that node or by other upstream nodes in its anomaly propagation set (or both). Any combination of $\boldsymbol{x}$ and $\boldsymbol{o}$ that satisfies all constraints in Equation (1) is a feasible explanation (cause) for the evidence observed by the sensors and satisfies the connection between these nodes in the corresponding BN. The second type of edges in this BN is related to the stochastic edges between network anomaly status and potential sensor measurements. To characterize this edge, we need to use a conditional probability distribution of $\boldsymbol{y}_n$ given x_n as defined in Section 3.2.4.

3.3.2. Anomaly detection with a BN

At decision epoch t, it is assumed that sensor data $\boldsymbol{Y}(t)$ are available for a subset of evidence nodes, denoted by $E(t)$, that is $\boldsymbol{Y}(t) = \{y_{nj} | n \in E(t), j \in J\}$. The rest of the nodes, denoted by $E'(t)$, either generate no sensor data or are subject to missing points. Thus, $N = E(t)\ \mathrm{U}\ E'(t)$. The anomaly detection problem can be mathematically defined through the posterior distribution of hidden variables $\boldsymbol{x}(t)$ and $\boldsymbol{o}(t)$ given $\boldsymbol{Y}(t)$ and the network topology G as

$$\Pr(o_1(t)\ldots, o_{|N|}(t), x_1(t), \ldots, x_{|N|}(t) | \boldsymbol{Y}(t), G). \tag{2}$$

The most likely status of the network is the combination of $[o_1(t)\ldots, o_{|N|}(t), x_1(t), \ldots, x_{|N|}(t)]$ that satisfies the causal relationships between variables as listed in Equation (1) and maximizes the joint posterior distribution of $\boldsymbol{o}$ and $\boldsymbol{x}$ in Equation (2). This problem is often referred to as the MPE in the context of BNs. One approach to find the MPE is to list all the scenarios (combination of feasible $\boldsymbol{x}(t)$ and $\boldsymbol{o}(t)$) and compute the joint posterior probabilities. Given that there are $|N|$ nodes in the hidden/explanation set and each node has only two possible values, we will have to compute $2^{|N|}$ conditional probabilities in total (although some combinations are not feasible). Such an exhaustive abductive inference is proved to be NP-hard in BNs (Cooper, 1990). In this article, we proposed an efficient way to evaluate Equation (2) and find the $K(K \ge 1)$ most likely scenarios for the status of the network and the uncertainty associated with that status while satisfying the topological dependencies

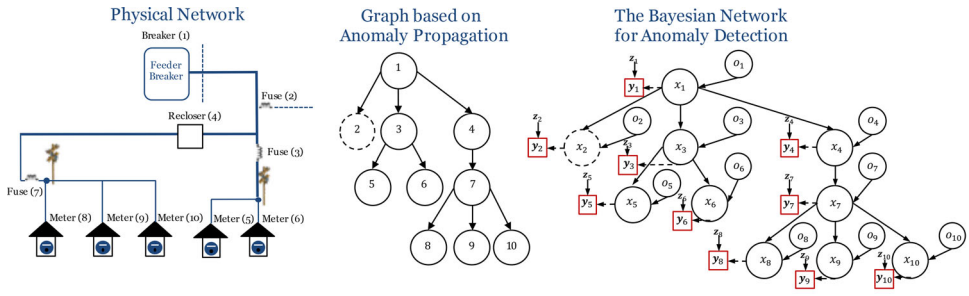

Figure 3. The BN transformation of a small section of a distribution network.

in the network. The equivalent binary optimization model for this problem can be formed as follows:

$$\max\ z = \quad \Pr(o_1(t), ..., o_{|N|}(t), x_1(t), ..., x_{|N|}(t) | \boldsymbol{Y}(t), G). \tag{3}$$

$$\text{Subject To}: \quad \Phi(G, \boldsymbol{x}, \boldsymbol{o}), \tag{4}$$

$$o_n(t) \in \{0, 1\}, \ \ x_n(t) \in \{0, 1\}, \ \ \forall n \in N, \tag{5}$$

where Φ is the set of constraints that guarantees the causal dependencies (or conditional relationships implied by the graph) between the elements $\boldsymbol{x}$ and $\boldsymbol{o}$ (Equation (1)).

3.4. A fast stochastic sampler for network anomaly detection

In this subsection, we discuss how to find the most likely network status given the data observed from a set of sensors across the network, denoted by $\boldsymbol{Y}(t)$. We refer to E as the set of nodes (evidence nodes) where at least one sensor measurement is available and E' as the set of nodes with no data (non-evidence node). Inspired by interesting properties of BNs, we first introduce an important remark that helps simplify the evaluation of Equation (2) based on the BN topology. For notational convenience, the time index t is removed from all equations.

Remark 1. Given that the network topology $G = (\mathbf{A}, \mathbf{P}, \mathbf{Z}, \mathbf{Y})$ is known, the following holds true:

$$\begin{aligned} &\Pr(o_1, ..., o_{|N|}, x_1, ..., x_{|N|} | \boldsymbol{Y}, G) = \Pr(o_1, ..., o_{|N|} | \boldsymbol{Y}, G) \\ &\propto \prod_{n \in N} \Pr(o_n) \times \prod_{n \in E} \Pr(\boldsymbol{y}_n | x_n, \boldsymbol{z}_n) \times \mathbb{1}\{(\boldsymbol{o}, \boldsymbol{x}) \in \Phi(G, \boldsymbol{o}, \boldsymbol{x}), \end{aligned}$$

where $\Pr(o_n)$ is the prior probability of node n being the anomaly source and $\mathbb{1}$ is the Dirac Delta function.

The proof is given in Section B of the Supplemental Materials. This Remark significantly simplifies the process of finding MPE based on the topology of the network. Also, this remark implies that it is sufficient to estimate $o_n, n \in N$ to find all $x_n, n \in N$ from Equation (1). Any combination of $\boldsymbol{x}$ and $\boldsymbol{o}$ that violates Equation (1) will result in the posterior values of zero and will not be the solution of the MPE. As will be discussed later, we will make sure that such solutions do not appear in the final solution sets. Below, we develop a Metropolis–Hastings sampling method to construct a Markov chain with a stationary distribution equal to Equation (2) according to Remark 1 with a block sampling of $\boldsymbol{x}$ and $\boldsymbol{o}$ variables while satisfying their causal relationships. The initial algorithm can sample o_i for node i for a number of predetermined iterations. A new value o_i^{new} is sampled from $Q(o_i^{new} | o_i^{old})$ where o_i^{old} is the current value of o. The acceptance probability can now be calculated as follows:

$$\alpha = \min\left\{1, \frac{Q(o_i^{old} | o_i^{new}) \ \ \Pr(o_i^{new} | \text{MB}(o_i))}{Q(o_i^{new} | o_i^{old}) \ \ \Pr(o_i^{old} | \text{MB}(o_i))}\right\}, \tag{6}$$

where $\text{MB}(o_i)$ is the current value of the nodes in the Markov Blanket (MB) of o_i. In a Bayesian framework setting, a MB of a node comprises a set of parents, children, and the parents of all its children. The main property of a MB is that a node is conditionally independent of all other nodes in the network given its MB. This helps the sampling process to randomly sample o_i conditioned on the previously generated values of the variables in the MB of o_i. For the proposal distribution Q, we use the model composition approach where we always flip the binary variable at each state, that is

$$\begin{aligned} &Q(o_i^{new} = 0 | o_i^{old} = 1) = Q(o_i^{new} = 1 | o_i^{old} = 0) \\ &= 1 \rightarrow Q(o_i^{new} | o_i^{old}) = Q(o_i^{old} | o_i^{new}). \end{aligned}$$

Now, $\Pr(o_i^{new} | \text{MB}(o_i))$ can be calculated from Remark 1 and the MB of node o_i in the BN, which are the parents of node o_i, the children of node o_i, and the parents of the children of node o_i. Given the deterministic relationship between the variables in the corresponding BN, the MB of node i includes a set of nodes that are shown in Figure 4. It can be seen that due to the deterministic relationship between nodes, the MB of node n involves many upstream nodes that can potentially impact this node. In Section 4.1.1, we discuss how to efficiently find the MB of all nodes. Now, the marginal distribution of o_i given its MB can be found as follows:

$$\begin{aligned} &\Pr(o_i^{new} | \text{MB}(o_i)) \propto \Pr(o_i^{new}) \\ &\times \prod_{n \in \{i, D_i\}} \Pr(\boldsymbol{y}_n | x_n^{new}, \boldsymbol{z}_n) \times \mathbb{1}\{(\boldsymbol{o}^{new}, \boldsymbol{x}^{new}) \in \Phi(G, \boldsymbol{o}, \boldsymbol{x})\}. \end{aligned}$$

It is important to note that $\boldsymbol{x}^{new}$ should be updated based on o_i^{new}. The key point here is to only update the elements in $\boldsymbol{x}$ that are impacted directly by o_i^{new}. From Equation (1) and the structure of the DAG in Figure 2, we know that o_i can impact x_i and all $x_n, n \in D_i$. Also, each x_i is initially impacted by their ancestors. Thus, selected elements of $\boldsymbol{x}$

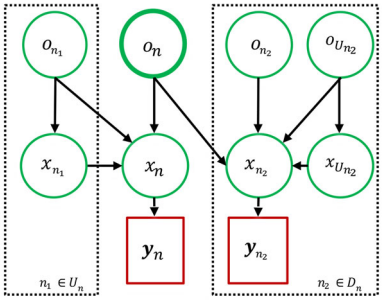

Figure 4. MB of node i based on deterministic and stochastic relationships.

should be updated based on o_i^{new} as follows:

$$x_n^{new} = \min\{1,\ o_i^{new} + \sum_{n' \in \{U_n\},\, n \neq i} o_{n'}^{update}\}, \quad n' \in \{i, D_i\}, \tag{7}$$

where $o_{n'}^{update}$ refers to the most updated value of $o_{n'}$ during the sampling process. The above equation is extracted from the topology constraints given in Equation (1). Given that only elements of o are sampled at each step and all x variables are calculated directly from o using Equation (1), the new combination of $\boldsymbol{o}^{new}$ and $\boldsymbol{x}^{new}$ always satisfies Equation (1) and, as a result, $\mathbb{1}\{(\boldsymbol{o}^{new}, \boldsymbol{x}^{new}) \in \Phi(G, \boldsymbol{o}, \boldsymbol{x})\}$ is always one. Now, the acceptance probability α can be simplified to

$$\alpha = \min\{1, \frac{\Pr(o_i^{new}) \times \prod_{n \in \{i, D_i\}} \Pr(\boldsymbol{y}_n | x_n^{new}, \boldsymbol{z}_n)}{\Pr(o_i^{old}) \times \prod_{n \in \{i, D_i\}} \Pr(\boldsymbol{y}_n | x_n^{update}, \boldsymbol{z}_n)}\}. \tag{8}$$

The summary of our proposed sampler is presented in Algorithm 1. At each iteration of the algorithm, variables $o_1, \ldots o_{|N|}$ are sampled sequentially and then $x_1, \ldots x_{|N|}$ are updated if necessary based on Equation (7). This means that even though one variable is sampled at a time, multiple hidden variables may be updated after a new sample is generated. It should be noted that the variables $\boldsymbol{x}^{update}$ and $\boldsymbol{o}^{update}$ are defined to track the most updated set of variables within a single iteration of MCMC. The stopping criterion is defined so that the algorithm is terminated when the variance in sample average for each variable in the last η iterations equals zero, that is, when the sample average remains the same for the last η iterations for each of the unknown variables o_i and x_i. Here, the stopping parameter η should be defined by the users.

Algorithm 1 Proposed Stochastic Sampler to Estimate Network Status Variables

Input: Network Structure (G), Evidence nodes (E), Sensor Data (Y), Burn-in Period (k_0), Stopping Criterion Parameters (η)

Output: Samples $\boldsymbol{o}^{(k)}$ and $\boldsymbol{x}^{(k)}$ for $k \in \{1, \ldots, K\}$

1: Initialize $\boldsymbol{o}^{(0)}$ from the corresponding priors and then find $\boldsymbol{x}^{(0)}$ from the network topology.
2: Set $\boldsymbol{x}^{update} = \boldsymbol{x}^{(0)}$ and $k = 1$.
3: **while** $k < \eta$ OR the variance of each sample average in the last η iterations is greater than 0 **do**
4: **for** $i = 1 : N$ **do**
5: - Propose $o_i^{new} = 1 - o_i^{(k-1)}$
6: - Compute the acceptance probability as
7:

$$\alpha = \min\left\{1, \frac{\Pr(o_i^{new}) \prod_{n \in \{i, D_i\}} \Pr(\boldsymbol{y}_n | x_n^{new}, \boldsymbol{z}_n)}{\Pr(o_i^{(k-1)}) \prod_{n \in \{i, D_i\}} \Pr(\boldsymbol{y}_n | x_n^{update}, \boldsymbol{z}_n)}\right\}$$

where

$$x_n^{new} = \min\left\{1,\ o_i^{new} + \sum_{j' \in \{U_n\},\, n \neq i} o_{j'}^{update}\right\}, \quad n \in \{i, D_i\} \tag{9}$$

8: - Sample a random number u from $\mathrm{U}(0, 1)$.
9: **if** $u \leq \alpha$ **then**
10: - Update $o_i^{(k)} = o_i^{new}$, $o_i^{update} = o_i^{new}$, $x_n^{(k)} = x_n^{new}$ and $x_n^{update} = x_n^{new}$ for $n \in \{i, D_i\}$
11: **else**
12: - Reject the proposed o_i^{new}, set $o_n^{(k)} = o_n^{update}$ and $x_n^{(k)} = x_n^{update}$ for $n \in \{i, D_i\}$.
13: **end if**
14: **end for**
15: **end while**

3.5. The most likely scenarios and their uncertainties

The outcome of the stochastic sampler is a set of samples $\boldsymbol{o}^{(k)} = [o_1^{(k)}, \ldots, o_{|N|}^{(k)}]$ and $\boldsymbol{x}^{(k)} = [o_1^{(k)}, \ldots, o_{|N|}^{(k)}]$ for $1 \leq k \leq K$, where K is the number of iterations that is either directly predetermined or indirectly imposed by a stopping criterion. Since the size of the networks in this study is typically large, there are many combinations of solutions that may not appear during the sampling process. Also, many solutions may appear only a few times throughout the entire MCMC. We can find the posterior distribution of each sample directly using Equation (2). In other words, only the posterior values of MCMC samples are evaluated. The index of the rth most likely sample can be found as:

$$I^r = \underset{k_0 \leq k \leq K,\, k \neq \{I^1, \ldots, I^{r-1}\}}{\arg\max} \Pr(\boldsymbol{O} = \boldsymbol{o}^{(k)}, \boldsymbol{X} = \boldsymbol{x}^{(k)} | \boldsymbol{Y}(t), G), \tag{10}$$

where k_0 is the burn-in period. Now, the rth most likely scenario would be $\mathrm{MPE}^r = [\boldsymbol{o}^{(I^r)}, \boldsymbol{x}^{(I^r)}]$. There are other ways to estimate the status of each node from the MCMC results. For instance, one can estimate the marginal expectation of the status of each node and its probability as follows:

$$\hat{o}_i = \mathrm{round}\left(\frac{\sum_{k=k_0:K} o_i^{(k)}}{K - k_0 + 1}\right) \text{or} \quad \mathrm{median}\{o_i^{(k_0)}, \ldots, o_i^{(K)}\},$$

$$\text{with probability } \frac{\sum_{k=k_0:K} \mathbb{1}\{o_i^{(k)} = \hat{o}_i\}}{K - k_0 + 1}.$$

The MCMC results can be used to rank nodes and regions (neighborhoods of nodes) based on their marginal distributions obtained from MCMC samples. For a complex function of interests where the closed-form formula for the posterior cannot be used, we directly use the outcome of the MCMC to get an estimate for that function. For instance, let us assume that the values for hidden nodes in set A are already known and we are only interested in knowing the

probability of the hidden status of nodes in set B. Then we can use

$$\Pr(\boldsymbol{o}_B|\boldsymbol{Y}, G, \boldsymbol{o}_A) \approx \sum_{k=k_0:K} \mathbb{1}\{\boldsymbol{o}_B \in \boldsymbol{o}^{(k)}\} \left[\sum_{k=k_0:K} \mathbb{1}\{\boldsymbol{o}_A \in \boldsymbol{o}^{(k)}\}\right]^{-1}. \tag{11}$$

The above equation is widely applicable due to its generic structure. It can estimate the probability of a certain set of nodes given that the sensor data for a certain group of sensors and the anomaly status of a certain set of nodes are already known. If one wants to calculate the rth most likely sample given that the status of nodes in set A are known, we need to update Equation (10) as follows:

$$I^r = \underset{k_0 \le k \le K,\, k \ne \{I^1, \ldots, I^{r-1}\} \& \boldsymbol{o}_A \in \boldsymbol{o}^{(k)}}{\arg\max} \Pr(\boldsymbol{O} = \boldsymbol{o}^{(k)}, \boldsymbol{X} = \boldsymbol{x}^{(k)}|\boldsymbol{Y}(t), G). \tag{12}$$

3.6. Theoretical bounds on sensor properties to detect anomalies

In this subsection, an important remark is introduced to help understand the power of sensor networks to detect anomalies in different sizes of subgraphs. The remark is shown for the case where there is only one type of sensor observation (i.e., $J = 1$), but it can be simply extended to the general case of $J > 1$.

Remark 2. For a network with sensors with $\alpha > \beta$, if node n is the true source of an anomaly and h denotes the actual fraction of the sensors located in the corresponding propagation subgraph that generates an error (i.e., $y = 1$), then for the model to detect node n as the source of the anomaly, h should satisfy the following:

$$h \ge \frac{\frac{\log\left(\frac{p_0}{1-p_0}\right)}{|E_n|} - \log\left(\frac{1-\beta}{1-\alpha}\right)}{\log\left(\frac{\beta}{\alpha}\right) - \log\left(\frac{1-\beta}{1-\alpha}\right)},$$

where p_0 is the prior probability of each node being the source of an anomaly (i.e., $\Pr(o_n = 1)$) and E_n and $|E_n|$ are the set and the number of sensors located in the propagation set of node n, respectively, that is $|E_n| = \sum_{n' \in \{n, D_n\}} z_{n'1}$. In the absence of knowledge for prior p_0, we can either define an uninformative prior or fine tune it with cross-validation. The proof of this remark and some useful insights are given in Section C of the Supplemental Materials.

4. Scaling up the anomaly detection process for large networks

The problem of network anomaly detection is often a large-scale problem with many nodes and large amounts of sensor data. That makes the MCMC sampler extremely slow to converge. In this article, we utilize two approaches, namely, parallelization and vectorization, to significantly accelerate the developed MCMC sampler and make it scalable for large networks.

4.1. Utilizing the topology of the BN for parallel sampling

A typical technique to conduct parallelization in MCMC algorithms is to run multiple MCMC chains independently in parallel (and potentially on multiple CPU units) and merge the results. Although many reports of success with such a parallel MCMC have been reported in numerous application settings, such an approach does not impact the computational time of each iteration of the MCMC algorithm and may not work efficiently for problems with many variables and slow convergence. In this article, we propose to decompose the proposed MCMC frameworks to structural segments/clusters and achieve chain parallelism by efficiently running one chain per available processing unit at a time. It is clear that all variables cannot be sampled in parallel at the same time for many reasons. For instance, samples may become infeasible and non-ergodic, which means that a solution may not satisfy the topological dependencies between nodes and may never converge to the stationary distribution. In this article, we utilize the concept of MBs for MCMC parallelization, graph coloring (Wang *et al.*, 2017), and chromatic sampler (Gonzalez *et al.*, 2011) to propose a parallel MCMC for the BN under study in this article. It is clear that in a BN, two nodes can be sampled in parallel if neither node is part of the other's MB, that is, their conditional distributions do not depend on each other. This is mainly because any variable is conditionally independent of all other variables given its MB. Thus, if we find a set of clusters where within each cluster no node is part of the other nodes' MB, then we can sample all nodes in a cluster simultaneously. Ideally, we need to maximize the number of nodes within each cluster and minimize the number of clusters while making sure each node belongs to exactly one cluster. Once the MCMC runs in parallel for the nodes of one cluster, the results are transferred to the next cluster, and this process continues until the nodes in all clusters are sampled. It is important to note that such a parallel sampling by vectorization in this article takes a Single-Instruction-Multiple-Data form in which the logical processors perform a single instruction on multiple data points simultaneously (using built-in Python libraries). The parallel sampling utilized in our numerical experiment section is a single-chain strategy, but it can be further extended to a multi-chain strategy as well as other parallelization structures, such as Multiple Instruction Multiple Data both of which are beyond the scope of this article.

4.1.1. MBs and network topology

To incorporate the above idea into the sampling process, we first need to extract the MB of each node systematically and generate a pseudo-adjacency matrix $\mathbf{B} = [b_{nm}]_{|N|\times|N|}$, where the binary parameter b_{nm} denotes whether node m is in the MB of node n. The reason that we need to derive this matrix is twofold: the MB nodes of any node are not necessarily its immediate neighbors (i.e., not directly known as network inputs) and the pseudo-adjacency matrix can be better utilized for finding the clusters for parallelization.

Table 2. Relationship between hierarchy, propagation matrix, and Pseudo propagation matrix.

	Adjacency Matrix										*Propagation Matrix*										*Pseudo Adjacency Matrix*									
Node Number	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2		1										1									1	1								
3			1		1	1							1		1	1					1		1		1	1				
4				1			1							1			1	1	1	1	1			1			1	1	1	1
5					1										1						1		1		1					
6						1										1					1		1			1				
7							1	1	1	1							1	1	1	1	1			1			1	1	1	1
8								1										1			1			1			1	1		
9									1										1		1			1			1		1	
10										1										1	1			1			1			1

Using the definition of MB, b_{nm} for the BN in this problem can be defined according to the set of anomaly propagation paths as

$$b_{nm} = \begin{cases} 1 & \forall\ m \in \{n, U_n, D_n, U_{D_n}\}, \\ 0 & \text{Otherwise.} \end{cases} \tag{13}$$

Based on the above definition, the children, parents, and the parents of each node's children build the MB of any node. The above equation implies that o_n cannot be sampled together with o_i if node i is in the anomaly propagation of node n or if node n is in the anomaly propagation of node i. In other words, matrix $\mathbf{B}$ is an adjacency matrix where adjacency is defined based on whether two nodes are in the MB of each other. We can infer from the above equation that the pseudo-adjacency matrix can be computed by the propagation matrix from the following remark:

Remark 3. Given that the elements of $\mathbf{B}$ should be binary, the following relationship holds between $\mathbf{B}$ and propagation matrix $\mathbf{P}$:

$$\mathbf{B} = \min\{[\mathbf{1}]_{|N|\times|N|}, \mathbf{P} + \mathbf{P}^\top + \mathbf{P} \quad \mathbf{P}^\top\}. \tag{14}$$

In the above equation, matrix $\mathbf{P}$ captures all children, matrix $\mathbf{P}^\top$ captures all parents, and matrix $\mathbf{P} \quad \mathbf{P}^\top$ captures the parents of all children. Since the MB relationship between two nodes may be counted more than once and given the binary nature of each element b_{nm} in matrix $\mathbf{B}$, we need to find the minimum of one and the outcome from $\mathbf{P} + \mathbf{P}^\top + \mathbf{P} \quad \mathbf{P}^\top$. It is important to note that the propagation matrix $\mathbf{P}$ indicates the propagation relationship between any two nodes in graph $\mathbf{G}$, which can represent whether an anomaly at any node n can propagate to any node m, where n and m are any two nodes in the network. Due to the deterministic relationship between node n and all its downstream nodes given in D_n, the anomaly of node n will propagate to all its downstream nodes. Therefore, propagation matrix $\mathbf{P} = [p_{nm}]_{|N|\times|N|}$ can be represented as follows:

$$p_{nm} = \begin{cases} 1 & \forall\ m \in \{n, D_n\}, \\ 0 & \text{Otherwise.} \end{cases} \tag{15}$$

Also, the propagation matrix can be generated from adjacency matrix $\mathbf{A}$ as follows:

$$\mathbf{P} = \min\{[\mathbf{1}]_{|N|\times|N|}, \mathbf{A}^\gamma\}, \tag{16}$$

where γ is the depth of graph G. The depth of a DAG is defined as the number of edges on its longest path between any two nodes. For a DAG with a source node (a vertex without incoming edges), depth is the longest distance from the source node to all other vertices in the given graph. In Table 2, the adjacency, propagation, and Pseudo adjacency matrices are given for a small network.

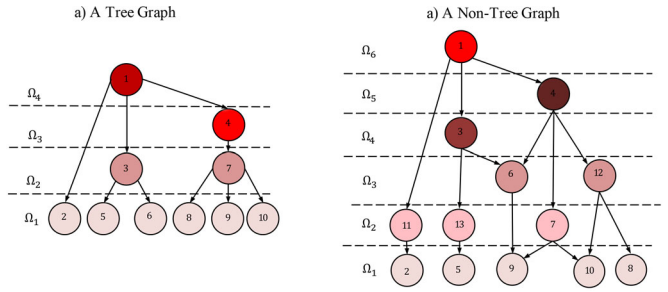

Figure 5. Examples of the node clustering on the tree network topology given in Figure 3 (a) and a random non-tree network structure (b).

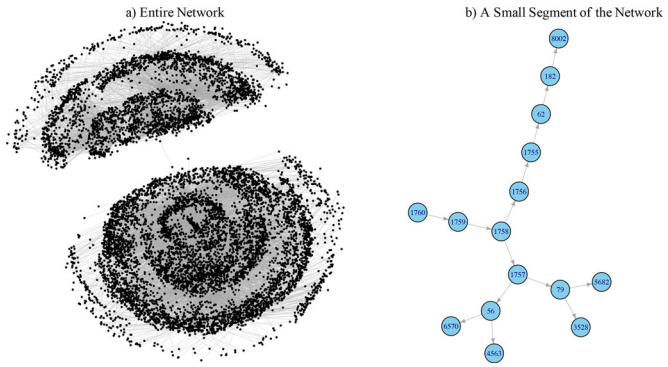

Figure 6. A simple view of the network used for numerical experiments (left), and the propagation path of an anomaly for a random subgraph (if an anomaly occurs at node 1760).

4.1.2. Finding clusters for parallel sampling

The next step is to find the clusters and variables within clusters that can be sampled together. To do that, we can borrow an idea from the well-known graph coloring problems, which have many applications in mathematics and computer science. In graph coloring problems, vertex coloring is conducted so that no two adjacent vertices share the same color. Considering the pseudo-adjacency matrix, we can use graph coloring to find the set of nodes that can be

sampled together (nodes that are not in the MB of others). We utilize a fast greedy approach inspired by Wang *et al.* (2017) that considers the unknown nodes of the BN based on a predefined sequence and then assign each node its first available color (cluster number). The idea is to first find the sink nodes, which are the nodes without outgoing edges (children), and then start from the one with the minimum degree. Once the independent sink nodes are chosen (with regards to their MB), they are removed from the network and the process continues until all nodes are assigned to the clusters. When arriving at a node, we check all existing clusters to verify if an assignment to them is possible. If not (because at least one of the nodes in each cluster is within the MB of the existing node), then we build a new cluster and move on. The details of the graph clustering for parallelization are given in Algorithm 2. The outcome of this algorithm is C clusters, denoted by $\Omega_1, ..., \Omega_C$, and the variables in each cluster. It is clear that $\Omega_{c_1} \cap \Omega_{c_2} = \emptyset$ for any pair $(c_1, c_2) \in \{1, ..., C\}^2$ and $\Omega_1 \cup \Omega_2 ... \Omega_C = N$. This algorithm works for both tree and non-tree structures. Also, it is possible that there is more than one way to cluster a network. In Figure 5, we provide a simple example of a 10-node tree graph and a 13-node non-tree graph to show which nodes can be sampled together based on the algorithm. Based on this graph, there are a total of four clusters with 6, 2, 1, and 1 nodes for the tree graph and a total of six clusters with 5, 3, 2, 1, 1, and 1 nodes for the non-tree graph. The nodes within each cluster can be sampled together. It is interesting to note that in the context of power distribution networks, devices of the same type can be clustered together, as they are not within each other's MB. For instance, all meter nodes can be sampled simultaneously, then transformer nodes can be sampled together, and so on (see Figure 1 in Supplemental Materials).

Algorithm 2 Graph Clustering for MCMC Parallelization for Tree and Non-Tree DAGs

Input: Network Structure (G)
Output: Clusters $\Omega_1, ..., \Omega_C$

A. Initialization:
– Find the Pseudo-Adjacency Matrix from

$$\mathbf{B} = \min\{[\mathbf{1}]_{|N|\times||N}, \mathbf{P} + \mathbf{P}^\top + \mathbf{P} \quad \mathbf{P}^\top\}.$$

– Find the sink node with the minimum degree (denote it by e^*) and assign it to cluster 1 (i.e., $\Omega_1 = \{e^*\}$). In the case of a tie, choose one randomly.
– Set $C = 1$ and the set of unassigned nodes as $\mathcal{U} = \{1, .., N\} - e^*$.
B. Clustering:
for $j = 1 : |N| - 1$ **do**
– Find sink nodes and their degrees.
– Find the sink nodes with the minimum degree (e^*) from $\mathcal{U}$. In case of a tie, choose one randomly.
– Define $N_B(e^*)$ as the set of nodes in the MB of node e^*, that is

$$N_B(e^*) = \{n \in N; b_{e^*n} = 1\}.$$

– Assign e^* to the lowest possible cluster denoted by c^*, where

$$c^* = \min\{c \in \{1, ..., C\} | \Omega_c \cap N_B(e^*) = \emptyset\}.$$

In the case of a tie, assign e^* to the cluster with the highest number of nodes from the same type (if applicable).
if $c^* > 0$ **then**
– Update $\Omega_{c^*} = \Omega_{c^*} \cup e^*$
else
– Set $C = C + 1$ and create a new cluster $\Omega_C = e^*$.
end if
– Remove e^* from $\mathcal{U}$ (i.e., $\mathcal{U} = \mathcal{U} - e^*$).
end for

Once the clusters for parallel sampling are determined from the greedy approach, we can use a Chromatic sampler to sample variables in the same clusters and then process the clusters sequentially. For the problem discussed in this article, we make sure that the graph clustering algorithm considers the topological structure of the network and the types of nodes. In the cluster assignment step of the node clustering scheme (Algorithm 2), we make sure that nodes of the same types are clustered together to the extent possible. For instance, when there is a tie between two or more clusters, a node is assigned to the cluster with the highest number of nodes from the same type. Also, by first assigning all sink nodes to clusters, it is very likely that nodes of the same type are clustered together. These steps are important for the interpretability of clusters, particularly for power distribution networks. For instance, all smart meters in a power distribution network can be clustered together. By applying the steps in Algorithm 2, we first found the clusters of nodes such that all nodes in a given subset are not in each other's MBs. We then sampled each cluster in parallel and then merged the results.

4.2. Vectorization

After the parallelization clusters are found, the MCMC operations within each cluster require many repetitive steps (e.g., finding α) for many nodes at the same time. One approach that can utilize parallel operations on a CPU is vectorization that can potentially lead to the elimination of many *for* loops. In this subsection, we propose a series of steps to convert the proposed MCMC operations on a set of nodes into efficient and high-performance vectorized operations and matrix operations, which are supported by a typical CPU. For large networks and when GPU units are available, we can potentially speed up the MCMC even further. To further accelerate the convergence of the MCMC, we can also run multiple chains on several CPU units. Recall Ω_c as the set of nodes in the cth cluster where they can be sampled together. Below, we introduce a series of vectorization steps for the proposed MCMC sampler. The target of the first step of vectorization is to generate vector $\boldsymbol{\alpha}_{\Omega_c} = [\alpha_i; i \in \Omega_c]$, where α_i is the acceptance probability in the Metropolis–Hastings algorithm as follows:

$$\alpha_i = \min\left\{1, \frac{\Pr(o_i^{new}) \times \prod_{n\in\{i,D_i\}} \Pr(\boldsymbol{y}_n|x_n^{new}, \boldsymbol{z}_n)}{\Pr(o_i^{(k-1)}) \times \prod_{n\in\{i,D_i\}} \Pr(\boldsymbol{y}_n|x_n^{update}, \boldsymbol{z}_n)}\right\}.$$

Given that the proposal distribution in the algorithm flips the current variable values, we have

$$\boldsymbol{o}_{\Omega_c}^{new} = [1, 1, ..., 1]^\top - \boldsymbol{o}_{\Omega_c}^{(k-1)},$$

which is the vector of the proposed values for all the variables in the cth cluster. By taking the logarithm of the second term of α_i, we get

$$\log\left(\Pr(o_i^{new})\left[\Pr(o_i^{(k-1)})\right]^{-1}\right) + \sum_{n\in\{i,D_i\}} \log\left(\Pr(\boldsymbol{y}_n|x_n^{new}, \boldsymbol{z}_n)\right) - \log\left(\Pr(\boldsymbol{y}_n|x_n^{update}, \boldsymbol{z}_n)\right),$$

which can be vectorized for all the variables in Ω_c as follows:

$$\boldsymbol{\lambda}_{\Omega_c} = \begin{bmatrix} \cdots \\ (o_i^{new} - o_i^{(k-1)}) \\ \cdots \end{bmatrix}_{|\Omega_c|\times 1} \odot \begin{bmatrix} \cdots \\ \log\dfrac{\Pr(O_i=1)}{\Pr(O_i=0)} \\ \cdots \end{bmatrix}_{|\Omega_c|\times 1} + \mathcal{X}_{\Omega_c} \times \left[\begin{bmatrix} \cdots \\ (x_n^{new} - x_n^{update}) \\ \cdots \end{bmatrix}_{|\mathcal{D}_{\Omega_c}|\times 1} \odot \begin{bmatrix} \cdots \\ \log\dfrac{\Pr(\boldsymbol{y}_n|x_n=1,\boldsymbol{z}_n)}{\Pr(\boldsymbol{y}_n|x_n=0,\boldsymbol{z}_n)} \\ \cdots \end{bmatrix}_{|\mathcal{D}_{\Omega_c}|\times 1}\right]. \quad (17)$$

where $\odot$ is the element-by-element product operator, known as the Hadamard product. Here, matrix $\mathcal{X}_{\Omega_c} = [\pi_{ij}]_{|\Omega_c|\times|\mathcal{D}_{\Omega_c}|}$ is a binary matrix where its ijth element is

$$\pi_{ij} = \begin{cases} 1 & \text{if} \quad \mathcal{D}_{\Omega_c}(j) \in \{\Omega_c(i), D_{\Omega_c(i)}\}, \\ 0 & \text{Otherwise.} \end{cases}$$

where $\mathcal{D}_{\Omega_c}(j)$ is the jth element of $\mathcal{D}_{\Omega_c}$ and $\Omega_c(i)$ is the ith element of the cluster Ω_c. It is interesting to point out that the only elements of Equation. (17) that need to change during each step of the MCMC are vectors $[(o_i^{new} - o_i^{(k-1)})]_{|\Omega_c|\times 1}$ and $[x_n^{new} - x_n^{update}]_{|\mathcal{D}_{\Omega_c}|\times 1}$ and all other vector/matrix elements are constant and need to be computed just once before the start of the MCMC (like π_{ij}) as a vector. To compute x^{new} as a vector, we have

$$\boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{new} = \min\{[\mathbf{1}]_{1\times|\mathcal{D}_{\Omega_c}|}, [o_1, ..., o_{|N|}] \times \mathbf{P}_{\mathcal{D}_{\Omega_c}}\}, \quad (18)$$

where

$$o_i = \begin{cases} o_i^{new} & \text{if} \quad i \in \Omega_c \\ o_i^{update} & \text{Otherwise} \end{cases},$$

and $\mathbf{P}_{\mathcal{D}_{\Omega_c}}$ is a matrix of size $|N| \times |\mathcal{D}_{\Omega_c}|$, which includes only the selected columns of the anomaly propagation matrix corresponding to the elements of $\mathcal{D}_{\Omega_c}$. Note that since $[o_1, ..., o_{|N|}] \times \mathbf{P}_{\mathcal{D}_{\Omega_c}}$ gives the anomaly status of each node, it can potentially have values more than one, which reflects a rare case where a node is under the impact of more than one anomaly. That is why we get the minimum of one and each element of the product to get the binary status of each node in Ω_c in Equation (18). Now, by generating $|\Omega_c|$ random numbers from uniform distribution $U(0, 1)$, denoted by $\boldsymbol{u}_{\Omega_c}$, the new samples in a vector form can be calculated as

$$\boldsymbol{o}_{\Omega_c}^{(k)} = \boldsymbol{o}_{\Omega_c}^{(k-1)} + \left[\mathbb{1}\{\boldsymbol{u}_{\Omega_c} \le \alpha_{\Omega_c}\}\right]_{|\Omega_c|\times 1} \odot \left[\boldsymbol{o}_{\Omega_c}^{new} - \boldsymbol{o}_{\Omega_c}^{(k-1)}\right]. \quad (19)$$

Also, all the x variables need to be updated based on the following vectorized equation:

$$\boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{k} = \min\{1, \left[\cdots, o_i^{(k)}, ...\right] \times \mathbf{P}_{\mathcal{D}_{\Omega_c}}\}. \quad (20)$$

Note that $\boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{update}$ and $\boldsymbol{o}_{\Omega_c}^{update}$ need to be calculated similar to Equations (18)-(19), respectively. The final algorithm after applying the parallelization and vectorization is given by all details in Algorithm 3. Each iteration of the MCMC can now run very efficiently with vectorized operators. The order of clusters within each iteration of MCMC is from the lowest to the highest. Note that all time-consuming steps are in Phases I and II, which are conducted offline and only once for a network.

Algorithm 3 The Fast MCMC Sampler for Anomaly Detection

Input: Network Structure (G), number of iterations (K)/ stopping criterion, Sensor data from evidence nodes (Y)
Output: Clusters $\Omega_1, ..., \Omega_C$ and Samples $\boldsymbol{o}^{(k)}$ and $\boldsymbol{x}^{(k)}$ for $k \in \{1, ..., K\}$
Optional Steps:
- **Run Multiple Independent Chains on Multiple Processors**
- **Divide Each Cluster to Sub-Clusters and Run Each Sub-Cluster on a Single Processor**

I. Finding Conditionally Independent Clusters
- Find the Pseudo-Adjacency Matrix as follows:

$$\mathbf{B} = \min\{[\mathbf{1}]_{|N|\times||N}, \mathbf{P} + \mathbf{P}^\top + \mathbf{P} \quad \mathbf{P}^\top\}.$$

- Find clusters $\{\Omega_1, ..., \Omega_C\}$ from Algorithm 2.

II. Initialization
- Compute Matrix $\begin{bmatrix} \cdots \\ \log\dfrac{\Pr(\boldsymbol{y}_n|x_n=1,\boldsymbol{z}_n)}{\Pr(\boldsymbol{y}_n|x_n=0,\boldsymbol{z}_n)} \\ \cdots \end{bmatrix}_{|\mathcal{D}_{\Omega_c}|\times 1}$ and $\begin{bmatrix} \cdots \\ \log\dfrac{\Pr(o_n=1)}{\Pr(o_n=0)} \\ \cdots \end{bmatrix}_{|\Omega_c|\times 1}$ for $c \in \{1, ..., C\}$.
- Initialize $\boldsymbol{o}^{(0)}$ from priors and then find $\boldsymbol{x}^{(0)}$ from the network topology.
- Set $\boldsymbol{o}^{update} = \boldsymbol{o}^{(0)}$ and $\boldsymbol{x}^{update} = \boldsymbol{x}^{(0)}$.
- Define $\boldsymbol{o}_{\Omega_c}^{(0)}$, $\boldsymbol{x}_{\Omega_c}^{(0)}$, $\boldsymbol{o}_{\Omega_c}^{update}$ and $\boldsymbol{x}_{\Omega_c}^{update}$ based on the defined clusters $\Omega_1, ..., \Omega_C$.

III. Chromatic MCMC Sampling
for $k = 1 : K$ (or while the stopping criterion is not satisfied) **do**
  for $c = 1 : C$ **do**
    - **Vectorized MCMC**

– Find $\boldsymbol{o}_{\Omega_c}^{new}$ as follows:

$$\boldsymbol{o}_{\Omega_c}^{new} = [1, 1, ..., 1]^\top - \boldsymbol{o}_{\Omega_c}^{(k-1)}.$$

– Compute the acceptance probability vector $\boldsymbol{\alpha}_{\Omega_c}$ from Equation (17) and $\boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{new}$ from Equation (18).
– Sample $|\Omega_c|$ random numbers, denoted by $\boldsymbol{u}_{\Omega_c}$.
– The new samples in a vector form can be calculated as

$$\boldsymbol{o}_{\Omega_c}^{(k)} = \boldsymbol{o}_{\Omega_c}^{(k-1)} + \left[\, \mathbb{1}\{\boldsymbol{u}_{\Omega_c} \le \alpha_{\Omega_c}\} \,\right]_{|\Omega_c|\times 1} \odot \left[\boldsymbol{o}_{\Omega_c}^{new} - \boldsymbol{o}_{\Omega_c}^{(k-1)}\right],$$
$$\boldsymbol{o}_{\Omega_c}^{update} = \boldsymbol{o}_{\Omega_c}^{(k)},$$
$$\boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{(k)} = \min\{1, \left[\cdots, o_i^{(k)}, ...\right] \times \mathbf{P}_{\mathcal{D}_{\Omega_c}}\},$$
$$\boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{update} = \boldsymbol{x}_{\mathcal{D}_{\Omega_c}}^{(k)}.$$

end for

– Merge the result for clusters $\Omega_1, ..., \Omega_C$ and store $\boldsymbol{o}^{(k)} = [\boldsymbol{o}_1^{(k)}, ..., \boldsymbol{o}_N^{(k)}]$ and $\boldsymbol{x}^{(k)} = [\boldsymbol{x}_1^{(k)}, ..., \boldsymbol{o}_N^{(k)}]$.

end for

5. Numerical experiments

In this section, we validate our framework by performing numerical experiments for network anomaly detection on (i) a part of a synthetic power distribution network in the San Francisco Bay Area and (ii) a comprehensive set of randomly generated graphs. The first data set is from the U.S. National Renewable Energy Lab (NREL) (Krishnan *et al.*, 2020). First, we briefly describe the data set and explain how we set up our numerical experiments. We first used Algorithm 2 developed in Section 4.1 to obtain the set of clusters for efficient parallel sampling. Then we employed Algorithm 3 demonstrated in Section 4 to construct the estimation chain of variables o_n and x_n for all nodes in the network. Some important performance measures are reported to show the effectiveness of our proposed framework.

5.1. Experiments on power distribution graphs

We considered a subnetwork from the topology dataset, which contains 10,090 nodes in total. The network has only one source node and 8330 sink nodes. It is assumed that one sensor exists on each node and each sensor has only one binary attribute. A simple view of the network is given in Figure 6.

The adjacency matrix $\mathbf{A}$ is generated from the original node and edge list and then the propagation matrix $\mathbf{P}$ is found from Equation (16) in Section 4.1. A total of 500 time instances (network samples) are considered where anomalies are detected at each time instance independently of others. For each network sample, we randomly choose a node to be the source of an anomaly and then mark all nodes in its anomaly propagation set as anomalous. Whether an anomaly occurs at one node follows the preset prior probability of 0.0001 ($\approx 1/N$). It is assumed that the prior probabilities are the same for all nodes in the network (e.g., $\Pr(o_n = 1) = 0.0001, \forall n \in N$). We then simulated sensor data for all nodes based on the structure of the network and the status of each node. We assumed that binary sensor attributes are collected at each node. It is assumed that the sensor attributes at node n follow a conditional Bernoulli distribution given the status of node n as follows:

$$\Pr(y_n = 1 | x_n = x) = \alpha^x \beta^{1-x},$$

where α and β are the true positive rate and false positive rate of sensor outputs, respectively. Four different scenarios are considered for sensor behavior, case 1 with $\alpha = 0.9$ and $\beta = 0.1$, case 2 with $\alpha = 0.8$ and $\beta = 0.2$, case 3 with $\alpha = 0.7$ and $\beta = 0.3$, and case 4 with $\alpha = 0.6$ and $\beta = 0.4$. The simulated set of y_i values was used as the observation evidence to estimate o_i and x_i. In addition to the results presented here, we also discussed the clustering effect and the MCMC convergence on Sections D and E of the Supplemental Materials, respectively.

5.1.1. Results on anomaly detection

We selected six important and reasonable performance metrics to evaluate the performance of the proposed framework. These metrics are the false positive rate (false alarm rate or FPR), which indicates the percent of healthy nodes incorrectly detected as anomalous; true positive rate (detection rate, sensitivity, recall, or TPR), which indicates the percent of anomalous nodes that have been properly detected; precision (positive predictive value), which indicates the percent of correct anomalies out of the total number of reported anomalies; f-score, which is used as a general overview of the performance of the models; and overlap coefficient, which is a common measure for assessing the efficiency of network anomaly detection models in detecting real anomalies and measures the agreement between the affected and detected nodes (1 means perfect agreement and 0 means no agreement). This last metric is computed as the ratio of true positives and the summation of the true positives, false negatives, and false positives. In addition to the above measures,

Table 3. Performance measures of the proposed framework versus the optimal approach.

Model	*Sensor Properties*		*FPR*	*TPR*	*Precision*	*F-score*	*Overlap Coefficient*	*Source Detection Accuracy*	*CPU Time*
	α	β							
Optimal Values	0.9	0.1	0.0000	0.997	0.998	0.998	0.995	0.900	223.4
	0.8	0.2	0.0001	0.989	0.993	0.991	0.982	0.718	224.1
	0.7	0.3	0.0004	0.976	0.970	0.973	0.947	0.472	223.3
	0.6	0.4	0.0024	0.940	0.841	0.888	0.798	0.216	226.6
Proposed Model	0.9	0.1	0.0000	0.995	0.999	0.997	0.994	0.850	6.4
	0.8	0.2	0.0014	0.982	0.999	0.991	0.981	0.542	6.5
	0.7	0.3	0.0008	0.958	0.999	0.978	0.958	0.270	5.03
	0.6	0.4	0.0038	0.884	0.996	0.937	0.881	0.086	6.4

we report the CPU time and anomaly source detection rate as the percent of correct detection of the sources of anomalies. The baseline model is the direct optimization model discussed in Equations (3)-(5), which is designed to find the most likely anomalous segments in the network. The results are shown for 500 network samples in the testing set in Table 3 for four combinations of α and β.

The results in the table can be summarized as follows: Overall, the framework performs well in terms of detecting anomalous nodes; however, the detection power significantly decreases with a decrease in sensor detection power (lower α and higher β). Second, although all performance measures related to the detection of anomalous nodes perform reasonably well, the source detection accuracy values are not very close to the optimal values. After further checking the results, we found two interesting observations for this poor performance. First, due to the nature of the grid distribution networks, many two-neighbor nodes share almost the same number of nodes/sensors in their anomaly propagation sets. In many cases, our model finds a source that is very close to the original source of the anomaly; thus, the source detection accuracy is negatively impacted by this property that many nodes share almost the same sensors. Second, the stopping criteria lead to the convergence of the model before reaching the optimal solution in many cases. The third important observation for all experiments is the CPU time, which in our model is significantly lower. Due to the iterative nature of our model and the effective use of vectorization, unlike the optimization model that cannot be run for larger networks with more complex structures, our model can be applied in many real large-scale network systems. It should be noted that the natures of the optimization model and the proposed model in this article are fundamentally different and that the CPU time for each model may be impacted by how they are coded, and what algorithm/solver is used for optimization. Thus, the reported CPU times should be interpreted with caution.

To further analyze the power of the proposed model in terms of detecting anomalous segments of the network, we separated the cases where the sources of anomalies are detected and not detected and then plotted the number of sensors in the corresponding anomalous subgraph and the fraction of active sensors (sensors with output one). It can be seen from Figure 7 that most undetected cases are small subgraphs with a small number of sensors. Also, the fraction of activated sensors for the cases of undetected sources is significantly lower than the fraction for detected cases (also higher variance for undetected cases can be seen). This further indicates the stochastic nature of the sensor networks. Depending on what percentage of network sensors is active in the anomalous subgraphs, the detection power of the model varies. This simply means that detecting anomalies with smaller impact (i.e., with fewer anomalous nodes) is harder than detecting large-scale anomalies. This is consistent with real distribution networks where single-node anomalies are usually hard to detect on their own and are often detected after the impacted customer notifies the control center (which may be hours after the start of the outage). One way to improve the performance of the model in small subgraphs is to use more powerful/accurate (high α and low β) sensors or increase the number of sensor attributes across the network.

We further discussed this aspect of the sensor networks in Section F of the Supplemental Materials.

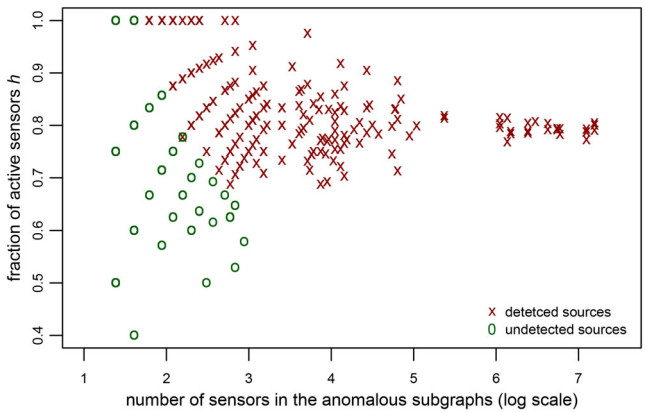

Figure 7. The relationship between the number of sensors, active sensors, and detection power.

5.2. Simulation experiments on random graphs

To evaluate the application of the proposed model with respect to size ($|N|$), number of attributes ($|J|$), power of sensors (represented by binary sensor outputs' true positive rate and false positive rate), and topology of the network for both tree and non-tree structures, we conducted a comprehensive set of experiments considering a total of 144 scenarios on randomly generated graphs as discussed below. A total of four types of well-known and widely used DAG topologies, where each has reportedly many real-world applications and are suitable for the study of complex networks,

Table 4. The average CPU time per network (seconds) for the proposed and benchmark models.

Approach	*N = 5000*			*N = 10,000*			*N = 20,000*		
	J = 1	*J = 10*	*J = 20*	*J = 1*	*J = 10*	*J = 20*	*J = 1*	*J = 10*	*J = 20*
Binary Classification	1	1	1	1	2	2	3	4	4
Proposed Approach	5	5	4	11	11	11	32	38	33
Optimal Approach	63	64	65	266	264	250	971	940	953
Optimal Graph Scan	14	12	15	244	253	303	5332	4653	8632

are considered. These four types of networks are regular DAG (a non-tree DAG graph where every node has about the same degree), Erdoes-Renyi (ER) DAG (a non-tree random graph that connects two nodes based on the edge probability or the edges selected uniformly at random), Watts Strogatz DAG (a non-tree graph that interpolates between the regular and ER graph), and tree/hierarchical DAG (a graph that follows a tree structure with no nodes having more than one parent). We considered four scenarios for the sensor properties as $\{\alpha = 0.9, \beta = 0.1\}$; $\{\alpha = 0.8, \beta = 0.2\}$; $\{\alpha = 0.7, \beta = 0.3\}$; and $\{\alpha = 0.6, \beta = 0.4\}$, where higher α and lower β represent more powerful sensors. A total of three scenarios for the number of sensor attributes (i.e., $|J| \in \{1, 10, 20\}$) and three scenarios for the number of nodes (i.e., $|N| \in \{5000, 10000, 20000\}$) are considered.

5.2.1. Performance evaluation

For each combination of the graph topology, number of nodes, number of features, and sensor power scenario, we simulated random anomalies in 100 independent graphs and applied our model to detect anomalies in the generated graphs. For each combination, we calculated all performance measures as discussed in Section 5.1. The results are summarized in Tables 4 and 5 given in the Supplemental Materials. Results verify that (as expected) as the number of features increases, the ability of the model to detect anomalies increases and all performance measures improve. This is simply because more information is available for each node of the network. For large values of $|J|$ (e.g., $J = 20$), the model performs almost perfectly, that is, almost all anomalous nodes and sources are detected accurately for scenarios 1-3 of sensor data detection power. When only one feature is available in the network, the model performs worse than cases with $|J| > 1$, but the performance improves as the sensor detection power increases (e.g., $\{\alpha = 0.9, \beta = 0.1\}$). The above observations apply to all graph topologies and graph sizes. In conclusion, more types of sensors are necessary, especially when sensors are not powerful (e.g., their detection power is low). Similarly, fewer sensors may be used when the detection power of sensors is high. Although the size of the network seems not to impact the detection power of the model, it does change the CPU time. However, the CPU time (even for the largest networks) is reasonable (less than 30 seconds for most configurations), which is sufficient for real-time decision making for systems such as power and water distribution networks. We found that for larger networks, it takes more iterations for MCMC to converge and that is why a few cases needed more than 1 minute to converge. This can be potentially improved by utilizing high-performance computing or running parallel chains of MCMC for faster convergence. As for the graph structure, it can be seen that the results for the tree graphs are slightly better than for non-tree graphs, particularly when there are less sensor features and fewer powerful sensors. Also, it takes the anomaly detection framework less time in tree networks to find anomalies, due to their simpler structures. Overall, the proposed anomaly detection model performs reasonably well and quickly for different graph structures, network sizes, features, and sensor detection power. However, for the cases with lower numbers of features and weaker sensor detection power, results are much worse than for the other cases. This makes sense as less powerful sensors provide less detection power and higher false alarms, which can only be compensated if more sensors are used. Another important factor that we found with a significant effect was the size of the anomalies (i.e., number of nodes impacted by anomalies), which was consistent with the results discussed earlier in Section 5.1.3. As expected, larger anomalies were easier to detect even for the cases where only one feature was used. We did not include here the results for how performance measures changed for different sizes of anomalies, because of page limits and the fact that a similar discussion was made before. Also, no consistent trend was found between the number of sensor outputs and model convergence and CPU time. We also studied the effect of having sensors of various types at each node and found no specific/interesting observation. We found that sensor effects are only based on their availability and detection power defined by the two parameters α and β. Due to page limits, we did not report the experimental results of having various sensor types. We have also reported experiments to evaluate the sensitivity of the model with respect to initialization, the impact of missing points and sensor availability, and the model's ability to detect zero-anomaly networks in Sections G, H, and I of the Supplemental Materials, respectively.

5.2.2. Comparison with potential benchmark models

In this subsection, we compare our work with three potential benchmarks models: the direct optimization model discussed in Equations (3)-(5), binary classification, and a modified version of the well-known GraphScan model discussed in Speakman *et al.* (2015). Although there are other available anomaly detection networks in the literature, none of them have the same assumptions and structure, thus, a direct and fair comparison would not be possible. We first briefly discuss these methods and how different they are from the proposed approach and then numerically show their performance.

- ***Direct Optimization***: In this model, Equations (3)-(5) are formed to build a binary linear programming problem. The solution of this model is the theoretically optimal values of network monitoring variables and can be used to show how close the results of our model are from their optimal values.
- ***Binary Classification***: In this class of models, the problem of network anomaly detection is formed as a node-level binary classification problem and then the trained model is used to label each node as normal or anomalous. The prediction results from all nodes will be integrated to capture the entire state of the network. We repeated the experiments using logistic regression, naive Bayes, support vector machine, and various forms of decision trees. Since logistic regression has the most consistent results across all experimental settings, we

reported only the results from logistic regression. For this experiment, we generated 100 extra networks just for training for each combination of setup parameters. Since the data are highly unbalanced through non-anomalous data, we used a weighted logistic regression to accommodate for the unbalanced nature of the data.

- ***Optimal GraphScan Model:*** The objective of the GraphScan model is to find the set of connected subgraphs with the highest anomaly scores without considering any propagation rules. This model is selected as it is the closest available model in terms of assumptions and objectives to our model, particularly because it aims to create the most likely set of anomalous subgraphs using binary node attributes. The two main inputs of this model are the anomaly score function and the neighborhood radius. We modified the model and transformed it into an optimization problem so that it also finds the optimal neighborhood size. Also, the modified version can handle multiple sensor attributes. The score of each node is calculated based on the common score function used in the GraphScan methods and some of its extensions as follows:

$$S_n(t) = y_n(t)\log\frac{\Pr(Y_n(t)=1|X_n(t)=1)}{\Pr(Y_n(t)=1|X_n(t)=0)} + (1-y_n(t))\log\frac{1-\Pr(Y_n(t)=1|X_n(t)=1)}{1-\Pr(Y_n(t)=1|X_n(t)=0)},$$

where $\Pr(Y_n(t)=1|X_n(t)=1)$ and $\Pr(Y_n(t)=1|X_n(t)=0)$ are the known and fixed true positive rate and false positive rate of the binary sensor attribute, respectively.

In all of our numerical experiments, each network is subject to only one anomaly. This is because the GraphScan model was not originally designed to find multiple anomalous subgraphs or the cases where there is no anomaly in the network. We conducted experiments for each of the 36 combinations of parameter setup based on N, J, α, and β. For each experiment, we simulated 100 networks for testing. Due to page limits, we only focus on regular networks and three key performance measures of overlap coefficient, source detection accuracy, and CPU time. We used the Rglpk package, which is a GNU solver based on an open source software *GLPK* developed for solving large-scale linear programming and mixed-integer linear programming. Our discussion below remains the same for other types of networks and performance measures. Results summarized in Figure 7 of the Supplemental Material verify that although the optimization-based models, perform the best in terms of detecting anomalies, our model performs very close to those models, particularly for larger values of sensor attributes and more powerful sensors. The binary classification model's performance is the worst, as it does not incorporate the topological dependencies between the nodes. The biggest gaps between our model and the optimal values are when the number of sensor attributes is one. We believe the gap can potentially become smaller by modifying the stopping criterion of the sampler and fine-tuning the initialization hyperparameter. To better understand the key benefit of our model compared with the benchmark models, we calculated the average CPU time per network for the task of anomaly detection as shown in Table 4. Since the results did not vary too much over different scenarios of sensor properties, results are averaged over the four cases of $\{\alpha, \beta\}$. It can be seen from the table that the binary classification model is the fastest, however, it cannot provide reasonable results at all. Our model is still very fast even for the network of size 20,000 nodes. Although the optimization-based models provide the best detection results, their long CPU times make them not scalable for larger networks and thus not always feasible in practice. About half of the CPU time for the optimization models is just to form the corresponding optimization models, which can potentially improve by forming them offline and using more advanced optimization solvers. However, even after that, the CPU time remains unreasonable for large networks. In addition to the above comparison, we should point out that our model outperforms these benchmark models from the following aspects: our model can be used to detect both normal networks (with no anomalous nodes) or networks with multiple anomalies. Also, due to the Bayesian nature of the proposed approach, the uncertainty of detection results can be utilized to provide more informed insights.

6. Concluding remarks

To monitor the status of large-scale networks and graph structures, many sensors can be installed at each node to collect partial information regarding each node's condition. These sensors are often not perfect and are subject to imperfect TPRs and non-zero FPRs. In this article, we develop an anomaly detection framework for heterogeneous attributed networks. This framework can integrate binary sensor data across the network and transform these data into optimal insights regarding the health status of the entire network. The problem of anomaly detection is formulated with a BN and then a stochastic sampling method is proposed to find the most likely status of the network. Effective parallelization and vectorization techniques are provided to make the detection problem scalable for large networks. In future work, we will investigate the dynamic nature of anomalies and sensor data and include more complex scenarios for anomalies in which the propagation sets are stochastic.

Funding

This article is based on work supported by National Science Foundation under Grant No. 1846975.

Notes on contributors

Feiran Xu is a doctoral student in Industrial and Systems Engineering at the University of Miami. Her research is focused on anomaly detection in large-scale sensor-driven networks. She holds a Master of Science in Global Logistics from W.P. Carey School of Business at

Arizona State University and a Bachelor of Management in Management Science from Sichuan University in Chengdu, China. Currently, she is a Graduate Teaching Assistant in the department and has presented her research at several conferences.

Ramin Moghaddass is an Associate Professor in the College of Engineering's Department of Industrial & Systems Engineering and the Director of the DOE Industrial Assessment Center and Data Analytics Lab. Dr. Moghaddass studies complex, sensor-driven engineered systems. His research is on developing a new generation of flexible and large-scale models for real-time system health monitoring inspired by dynamic structures and networks. His research is applicable in a number of engineering systems, including wind turbines, power systems, and other sensor-intensive engineering systems.

References

Akoglu, L., Tong, H. and Koutra, D. (2015) Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, **29**(3), 626–688.

Alghuried, A. and Moghaddass, R. (2020) Anomaly detection in large-scale networks: A state-space decision process. *Journal of Quality Technology*, **54**, 1–28.

Antonelli, D., Bruno, G. and Chiusano, S. (2013) Anomaly detection in medical treatment to discover unusual patient management. *IIE Transactions on Healthcare Systems Engineering*, **3**(2), 69–77.

Asadzadeh, P., Kulik L., Tanin E. and Wirth, A. (2011) On optimal arrangements of binary sensors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **6899**, 168–187.

Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. (2013) Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, **16**(1), 303–336.

Chandola, V., Banerjee, A. and Kumar, V. (2009) Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, **41**(3), 1–58.

Chen, J., Cao, K., Li, K. and Sun, Y. (2011) Distributed sensor activation algorithm for target tracking with binary sensor networks. *Cluster Computing*, **14**(1), 55–64.

Cooper, G.E. (1990) The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, **42**(2-3), 393–405.

Djuric, P.M., Vemula, M. and Bugallo, M.F. (2008) Target tracking by particle filtering in binary sensor networks. *IEEE Transactions on Signal Processing*, **56**(6), 2229–2238.

Gonzalez, J., Low, Y., Gretton, A. and Guestrin, C. (2011) Parallel Gibbs sampling: From colored fields to thin junction trees, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 11-13 April, Fort Lauderdale, FL, USA (vol. 15, pp. 324–332).

Guerriero, M., Willett, P. and Glaz, J. (2009) Distributed target detection in sensor networks using scan statistics. *IEEE Transactions on Signal Processing*, **57**(7), 2629–2639.

Hooi, B., Eswaran, D., Song, H.A., Pandey, A., Jereminov, M., Pileggi, L. and Faloutsos, C. (2019) GridWatch: Sensor placement and anomaly detection in the electrical grid. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11051, 71–86.

Janakiram, D., Kumar, A. V. U. P., & Reddy V. A. M. (2006). Outlier detection in wireless sensor networks using bayesian belief networks. In *2006 1st International conference on communication systems software & middleware*, New Delhi, India (pp. 1-6), doi: 10.1109/COMSWA.2006.1665221.

Kim, T.-Y. and Cho, S.-B. (2018) Web traffic anomaly detection using c-lstm neural networks. *Expert Systems with Applications*, **106**, 66–76.

Krishnan, V.K., Bugbee, B., Elgindy, T., Palmintier, B.S., Mateo, C., Postigo, F., Gomez San Roman, T., Duenas, P. and Lacroix, J.-S. (2020) Realistic synthetic distribution grids: Summary of validation results. NREL PR-5D00-75723.

Kwisthout, J. (2011) Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, **52**(9), 1452–1469.

Lacave, C. and Díez, F.J. (2002) A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review*, **17**(2), 107–127.

Li, Z. and D'Ambrosio, B. (1993) An efficient approach for finding the MPE in belief networks, in *Uncertainty in Artificial Intelligence*, Elsevier, pp. 342–349.

Marinescu, R. and Dechter, R. (2012) Best-first and/or search for most probable explanations. *arXiv preprint arXiv:1206.5268*.

Mengshoel, O.J., Wilkins, D.C. and Roth, D. (2010) Initialization and restart in stochastic local search: Computing a most probable explanation in Bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, **23**(2), 235–247.

Murugan, K. and Suresh, P. (2017, 02) Ensemble of ADA booster with SVM classifier for anomaly intrusion detection in wireless ad hoc network. *Indian Journal of Science and Technology*, **10**, 1–10.

Neapolitan, R.E., 2004. *Learning bayesian networks* (Vol. 38). Upper Saddle River: Pearson Prentice Hall

Noble, C.C. and Cook, D.J. (2003) Graph-based anomaly detection, in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, New York, NY, USA, pp. 631–636.

Ostfeld, A. et al. (2008) The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, **134**(6), 556–568.

Pearl, J. (1988) *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann.

Popat, R.R. and Chaudhary, J. (2018) A survey on credit card fraud detection using machine learning, in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, Tirunelveli, India, pp. 1120–1125.

Ranshous, S., Shen, S., Koutra, D., Harenberg, S., Faloutsos, C. and Samatova, N.F. (2015) Anomaly detection in dynamic networks: A survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, **7**(3), 223–247.

Speakman, S., McFowland, I. and Neill, D. (2015) Scalable detection of anomalous patterns with connectivity constraints. *Journal of Computational and Graphical Statistics*, **24**(4), 1014–1033.

Stojanovic, L., Dinic, M., Stojanovic, N. and Stojadinovic, A. (2016) Big-data-driven anomaly detection in industry (4.0): An approach and a case study, in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE Press, Piscataway, NJ, pp. 1647–1652.

Sy, B.K. (1992) Reasoning MPE to multiply connected belief networks using message passing, in *AAAI*, Taylor & Francis, pp. 570–576.

Tuptuk, N., Hazell, P., Watson, J. and Hailes, S. (2021) A systematic review of the state of cyber-security in water systems. *Water (Switzerland)*, **13**(1).

Wang, T.-C., Phoa, F. and Lin, Y.-L. (2017) Network exploration by complements of graphs with graph coloring. *Journal of Advanced Statistics*, **2**(2), 78–95.

Wang, Z., Bulut, E. and Szymanski, B.K. (2010) Distributed energy-efficient target tracking with binary sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, **6**(4), 1–32.

Yu, R., Qiu, H., Wen, Z., Lin, C. and Liu, Y. (2016) A survey on social media anomaly detection. *arXiv:1601.01102* **18**(1), 1–14.

Yuan, C. and Lu, T.-C. (2007) Finding explanations in Bayesian networks, in *The 18th International Workshop on Principles of Diagnosis*, DX-07, Nashville, TN, USA, pp. 414–419.

Yuan, Y., Dehghanpour, K., Bu, F. and Wang, Z. (2020) Outage detection in partially observable distribution systems using smart meters and generative adversarial networks. *IEEE Transactions on Smart Grid*, **11**(6), 5418–5430.

Zhang, Y., Chen, B. and Yu, L. (2020) Distributed fusion Kalman filtering under binary sensors. *International Journal of Robust and Nonlinear Control*, **30**(6), 2570–2578.