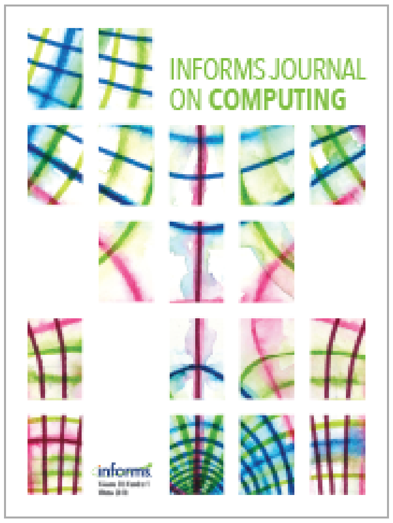


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Optimal Frameworks for Detecting Anomalies in Sensor-Intensive Heterogeneous Networks

Ramin Moghaddass, Yongtao Guan

http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–28
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Optimal Frameworks for Detecting Anomalies in Sensor-Intensive Heterogeneous Networks

Ramin Moghaddass,[a,b,*] **Yongtao Guan**[b]

[a] Department of Industrial and Systems Engineering, University of Miami, Coral Gables, Florida 33146; [b] Department of Management Science, Miami Herbert Business School, University of Miami, Coral Gables, Florida 33146
*Corresponding author

Contact: ramin@miami.edu, https://orcid.org/0000-0003-3634-633X (RM); yguan@bus.miami.edu (YG)

Received: August 20, 2020
Revised: May 6, 2021; October 11, 2021; February 7, 2022; February 25, 2022
Accepted: March 1, 2022
Published Online in Articles in Advance: May 10, 2022

https://doi.org/10.1287/ijoc.2022.1192

Abstract. Many network/graph structures are continuously monitored by various sensors that are placed at a subset of nodes and edges. The multidimensional data collected from these sensors over time create large-scale graph data in which the data points are highly dependent. Monitoring large-scale attributed networks with thousands of nodes and heterogeneous sensor data to detect anomalies and unusual events is a complex and computationally expensive process. This paper introduces a new generic approach inspired by state-space models for network anomaly detection that can utilize the information from the network topology, the node attributes (sensor data), and the anomaly propagation sets in an integrated manner to analyze the entire network all at once. This article presents how heterogeneous network sensor data can be analyzed to locate the sources of anomalies as well as the anomalous regions in a network, which can be impacted by one or multiple anomalies at any time instance. Experimental results demonstrate the superior performance of our proposed framework in detecting anomalies in attributed graphs.

Summary of Contribution: With the increasing availability of large-scale network sensors and rapid advances in artificial intelligence methods, fundamentally new analytical tools are needed that can integrate data collected from sensors across the networks for decision making while taking into account the stochastic and topological dependencies between nodes, sensors, and anomalies. This paper develops a framework to intelligently and efficiently analyze complex and highly dependent data collected from disparate sensors across large-scale network/graph structures to detect anomalies and abnormal behavior in real time. Unlike general purpose (often black-box) machine learning models, this paper proposes a unique framework for network/graph structures that incorporates the complexities of networks and interdependencies between network entities and sensors. Because of the multidisciplinary nature of the paper that involves optimization, machine learning, and system monitoring and control, it can help researchers in both operations research and computer science domains to develop new network-specific computing tools and machine learning frameworks to efficiently manage large-scale network data.

History: Accepted by Ram Ramesh, Area Editor for Data Science and Machine Learning.
Funding: This work was supported by the National Science Foundation CAREER Award [Grant 1846975].
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2022.1192.

Keywords: network analytics • anomaly detection • network monitoring • state-space models

1. Introduction

Many important systems in high-impact applications have a network/graph structure. Network systems or systems with a network/graph structure are subject to different types of anomalies, abnormal behavior, and disturbances over time. Anomalies in a network can be defined as a set of unusual behaviors or patterns that does not conform to a known normal behavior and can potentially impact one or multiple nodes. Examples are momentary or sustained outages in power distribution networks, distributed denial-of-service (DDoS) attacks in computer networks, and spam emails in social networks. A unique aspect of anomalies in a graph structure is that the impacted nodes are often connected or adjacent and may follow a set of topological constraints. For instance, in power distribution networks, if a transformer experiences a sustained anomaly, then all devices downstream from the affected transformer (such as customer meters) will experience a power outage. To monitor the health of modern networks and detect anomalies in a timely manner, different types of sensors and smart devices are installed across these networks that can track real-time data from a particular node or section of a network. With the increasing availability of large-scale sensors that monitor network entities, new analytical tools are needed that can integrate data collected from sensors across networks for decision making while taking into account the stochastic and topological dependencies between nodes, sensors, and anomalies. Available sensor-based models for network anomaly detection are rarely designed based on both the structure of the network and the data collected across the network. Also, networks include many dependent subsystems that are not the same in the sense that different subsystems may experience different types of anomalies and different sets of nodes may collect different types of sensor data.

Despite the growing importance of anomaly detection, this promising area has received relatively little attention for heterogeneous attributed networks. This article addresses important challenges for anomaly detection in sensor-intensive networks. The main challenges are finding the anomalous nodes or subgraphs and identifying the sources of anomalies given a known set of propagation rules between nodes and/or connectivity constraints. The next challenge is how to efficiently merge heterogeneous sensor data based on the topology of the network so that the healthy/normal and anomalous subgraphs are both detected throughout the network, which may be under the impact of multiple types of anomalies. In this article, we introduce new integrated frameworks for network data fusion and multiclass anomaly detection for attributed networks with a heterogeneous structure and disparate sensors. In the proposed integrated network data fusion model, each subnetwork with sensor data are transformed (compressed) into an embedded vector (in a low-dimensional space) that represents the dynamic of the subnetwork and its sensor data. The data in the embedding space are then used as inputs for a multilayer perceptron (MLP) structure to train a multiclass node-level anomaly detection model. Finally, two optimization models are introduced that can simultaneously find the location of anomalies and impacted nodes for the entire network. The outcomes of both optimization models are distinct subsets of network nodes (anomalous segments) such that all nodes in the subset share the same health or anomaly status. Because all nodes are considered in an integrated anomaly detection paradigm, it is expected that fewer false alarms are generated and, as a result, the true location of anomalies and impacted nodes are detected more accurately.

This paper is motivated by power distribution networks, which are sensor-intensive graph structures susceptible to various types of faults and anomalies that often lead to momentary or sustained power interruptions, costly power outages and repairs, and customer dissatisfaction. In such networks, the topological dependencies between nodes and edges make sensor data, potential anomalies, and impacted nodes across the network interdependent. For instance, anomalies experienced by a specific subset of connected nodes can form a connected anomalous subnetwork with similar sensor behavior at the individual node level. Thus, conventional approaches, such as typical machine learning models at the node level, cannot fully capture the underlying dependencies between the entities of the grid and are often ineffective in localizing anomalous regions in a network. Although this paper is motivated by distribution networks, it is general in terms of the network topology and can be used in a broad range of applications to detect unusual events. Examples are smart and connected cities (Parra et al. 2015), communication networks (Yang et al. 2011), and wireless sensor networks (Feng et al. 2017).

The contributions made in the paper are summarized as follows: An optimization framework is proposed that can detect an anomaly or multiple simultaneous anomalies in the network using observations from various types of sensors with multivariate outputs that are placed on a subset (or all) of the nodes. The detected anomalous nodes (subgraphs) are consistent with a known set of propagation paths (rules) between nodes and/or connectivity constraints. We also introduce network structure and sensor data aggregation and dimensionality reduction steps that help with improving the computational complexity and the accuracy of the anomaly detection task. This framework considers the topological dependencies between all nodes as well as the stochastic nature of sensor data. The framework is designed to consider the network all at once and to find the sources of anomalies as well as the subsets of impacted nodes with similar anomalous structures. Compared with many similar approaches, the proposed work has the advantage that it does not need user-specified input parameters (e.g., the neighborhood size/radius) to search for anomalies or anomaly score thresholds. Also, the proposed framework does not need any parametric probability distribution to model the stochastic behavior of sensor data. The network and the models proposed have generic structures, which allow for any network structure represented by a directed acyclic graph (DAG) and model family to be used for sensors and anomalies. Finally, all anomalies and impacted nodes/subgraphs are detected in an optimal manner; a set of important preoptimization steps are introduced to improve the scalability of the optimization models for anomaly detection. The frameworks utilized for dimensionality reduction and multiclass anomaly detection are formalized by well-known deep neural network structures, without any technical contributions to neural networks.

By employing deep autoencoder and multilayer perceptron, we aim to build a sparse model, removing uncorrelated variables falsely found to be correlated with anomalies while taking uncertainty and numerical instability into account. We should also point out that detecting changes/anomalies in the topology of the network is beyond the scope of the paper.

This article is organized into seven sections. Section 2 reviews some of the recent work on anomaly detection, particularly for network and graph structures. The fundamental structure of the network and its main elements are discussed in Section 3. In Section 4, the details of the proposed approach for network structure and sensor data aggregation and dimensionality reduction are presented. Then, a multilayer perceptron neural network is formalized for nodal anomaly detection. In Section 5, two optimization models are presented that can be used to detect anomalies throughout the network. Section 6 provides a comprehensive set of numerical experiments to show the application of the proposed frameworks. We conclude in Section 7 and suggest directions for future research.

2. Related Work

2.1. Anomaly Detection and Its Application to Networks and Graph Structures

Anomaly detection has been used in many applications, such as fraud detection (Lee et al. 2017), fault detection (Theissler 2017), flight safety prediction (Yelundur and Campbell 2013), network intrusion detection (Brice et al. 2011), and healthcare (Antonelli et al. 2013). However, traditional analytical methods for anomaly detection are that they are mostly general-purpose models or a byproduct of an algorithm designed for a purpose other than anomaly detection. As a result, they are not optimized to detect anomalies and may lead to too many false alarms or too few detected anomalies (Liu et al. 2012). A survey of the research on anomaly detection can be found in Liu et al. (2012) and Habeeb et al. (2019). Many systems have a network/graph structure that is susceptible to anomalies caused by a variety of off-nominal conditions. Anomaly detection for networks has been an important topic, but it has recently become even more popular because of the availability of new sources of structured graph data. For a comprehensive review of anomaly detection for network/graph structures, interested readers may refer to Ranshous et al. (2015) and Akoglu et al. (2015). Much of the available work on anomaly detection in network settings focuses mainly on searching individual nodes or sensors (Henderson et al. 2011). Monitoring and decision making based on a single sensor can result in information loss and can increase the false alarm rate (Wang et al. 2009). Multisensor information fusion can obtain more accurate and reliable information, which cannot be achieved by single-sensor analysis (Wang et al. 2009). Many of the available approaches involve general anomaly detection methods that fail to incorporate the complexities of networks and the interdependencies between network entities and sensors. For example, anomaly detection can be simply formulated as a classification problem with the unrealistic assumption that network data are independent and identically distributed and that the same types of sensor data are collected from all network nodes.

2.2. Network Topology and Sensor Data Aggregation

One of the modern techniques for analyzing graph data is to apply graph embedding, which can learn to generate a vector representation of graph topology and potentially node attributes. Most traditional network embedding approaches mainly focus on the network itself, ignoring attributes of nodes. In recent years, attributed network embedding methods that focus on graph topology and node features have received more attention (see, for instance, the recent work of Gao and Huang 2018 and Cui et al. 2020). However, these approaches have significant limitations: (i) they are focused only on node embedding and node classification as opposed to subgraph embedding and (ii) they cannot accommodate heterogeneous sensor outputs where only a selected set of nodes have sensors and sensor attributes are not the same across the network. Most available subgraph embedding approaches are fully supervised and are used for subgraph classification where the goal is to predict a label associated with a particular subgraph (Hamilton et al. 2017b). In Hamilton et al. (2017a), an unsupervised embedding is proposed; instead of training a distinct embedding vector for each node, a set of aggregator functions are used to aggregate feature information from a node's local neighborhood. However, that approach is only scalable for small neighborhoods. Despite its great potential, graph embedding on attributed networks for the task of anomaly detection has been rarely studied in the literature and has received attention only in the last few years. In this paper, we choose a scalable aggregation function that provides a standardized vector representation of any subgraph topology and sensor data. We will show that this aggregation function is injective with regard to detecting anomalies and is statistically sufficient (under certain conditions), that is, it captures all the information concerning the subgraph that is relevant for anomaly detection.

2.3. Techniques for Network Anomaly Detection

Techniques used for network anomaly detection can be classified into two categories: general-purpose models (e.g., classification and clustering) that may be marginally modified to be used for network anomaly detection and network-specific models that are mainly designed for network/graph structures. A comprehensive overview of various methods used in network anomaly detection is given in Bhuyan et al. (2014). The models and algorithms typically used for anomaly detection can also be classified into the categories of statistical models (e.g., control chart, histogram), residual-based models that are often parametric, clustering-based models, nearest neighbors, classification models, and context-specific models. These methods have unique strengths and weaknesses, and their performance often varies depending on the application. For example, strong distributional assumptions in statistical models; ineffectiveness in distance measures in methods such as clustering and nearest neighbor, particularly in high-dimensional settings; high vulnerability of unknown or modified anomalies in supervised classification models; and high false alarm rates and difficulty in model training in unsupervised models make these methods generally ineffective for complex graph structures. Another limitation of these approaches is that anomalies are treated very generally, and their cause and type are unidentifiable.

Models and frameworks for attributed networks that are specifically designed to work with network/graph structures and data/attributes have gained more attention over the past few years. In such frameworks, the main question is given a network/graph with node attributes, what nodes/subnetworks are anomalous? A comprehensive overview of the state-of-the-art methods for anomaly detection in graph data are given in Akoglu et al. (2015). Unlike conventional techniques, recent work on anomaly detection for attributed networks considers both network structure and nodal attributes (Ding et al. 2019). These models mainly find anomalous nodes whose behavior deviates significantly from the majority of reference nodes. Ding et al. (2019) modeled the attributed networks with a graph convolutional network and compressed the input attributed network to low-dimensional embedding representations. Then, the topological structure and nodal attributes were reconstructed by decoder functions. Finally, the reconstruction errors of nodes following the encoder and decoder were used to rank anomalous nodes. In Perozzi and Akoglu (2016), both attributes and network structure are utilized to analyze the abnormality of each node from the ego-network point of view and detect anomalous neighborhoods. Li et al. (2017) proposed a residual-based method that detects anomalies whose behavior is different from the majority by characterizing the residuals of attribute information and its coherence with network information.

Although our proposed approach is similar to the above-referenced works because we also utilize network structure and node attributes, the proposed approach is unique with respect to the following aspects. In addition to ranking all nodes and subnetworks, the proposed approach can figure out whether there is an anomaly (or multiple anomalies) and then find the sources of anomalies and impacted nodes. None of the above models can determine how many anomalies are in the network. Also, our model considers a heterogeneous network where a node may have no attribute or have its own set of attributes that is different from that of other nodes. Similarly, the set of sensors and attributes in two subnetworks may be different. Thus, we cannot simply compare two nodes or two neighborhoods and provide a ranking. Also, our framework can incorporate multiple types of anomalies and anomaly propagation sets for each node. Our work is different from community-based models and compression-based models used in dynamic networks (Ranshous et al. 2015) that are only based on the change or evolution in edges and nodes over time. Another group of commonly used anomaly detection models is distance-based methods in which a specific form of distance measures is employed as a metric to measure change or anomalies in the network. These types of models require a fixed number of attributes or predictors, which is not the case for heterogeneous attributed networks. Also, their success highly depends on the type of data, distance measures, and the dimensionality of the data (Gogoi et al. 2011). Because of its flexible structure, our framework can handle the above drawbacks of distance-based models.

Our work is also different from Liu et al. (2010) that considers a dynamic structure with moving objects in which a small subset is the sensor set with complete knowledge, and the status of the rest should be inferred from these sensors. Unlike this work, the topology of the network and the relationship between nodes defined by edges have a significant impact on the propagation of anomalies. Also, our sensors are noisy and only partially provide information regarding the status of each node. Our work can consider multidimensional node attributes and multiple simultaneous anomalies, both of which are beyond the scope of Liu et al. (2010). Our work is also different from many heterogeneous anomaly detection methods, such as Liu et al. (2018) and Moshtaghi (2013), because they cannot incorporate the dependencies and causal relationships between nodes that drive the structural propagation of anomalies across the network. Unlike the work of Liu et al. (2018), anomalies in our work are considered for a directed graph and are not defined based on any specific features. In fact, the propagation of anomalies follows a structural pattern (e.g., connected nodes or downstream nodes). The proposed approach in Moshtaghi (2013) cannot incorporate the propagation of anomalies and the dependencies between nodes across the network, and thus their detected anomalous nodes may or may not be connected. Also, that work relies on the selected similarity or distance measure; and sensor measurements under normal conditions are assumed to follow a Gaussian mixture density. Our work does not require such distributional assumption for normal data and can be used to detect structural anomalies in the network based on heterogeneous sensor data that may be collected only from a subset of nodes across the network.

2.4. Subset Scan Statistics Methods

Subset scan statistics approaches are one of the most widely used methods for detecting anomalies in graph data. These methods also seek to find anomalous patterns by filtering through subsets of subgraphs and identifying the subset with the highest anomaly score using a heuristic or an exact model. Many of the works in this domain focus on the detection of arbitrarily shaped connected clusters or specific shapes (Kulldorff 1997). The work in this domain can be divided into two categories: topology-based and sensor-based methods. Because our framework utilizes both network sensor data and sensor topology, it is different from topology-based methods, such as Priebe et al. (2005), that search for a highly anomalous area only based on the topology of the network. The work in the second category often utilizes binary sensor data and the network topology for anomaly detection. Examples are the well-known heuristic method of FlexScan (Tango and Takahashi 2005) and the exact method of GraphScan (Speakman et al. 2015). In both of these methods, the connected subgraph with the highest anomaly from among all the connected subgraphs is found based on observed counts from network sensors.

Although there are similarities between the proposed framework and the abovementioned references, there are key differences that make our work different from the above series of work. For instance, both the FlexScan and GraphScan approaches utilize a one-dimensional binary sensor output (score) to find a single subgraph of connected nodes that maximizes a score function over all feasible subgraphs. However, our approach is more generic and aims to find a subset or multiple subsets of connected nodes (as opposed to one subset) satisfying anomaly propagation paths by analyzing single or multidimensional binary sensor data collected from a subset of nodes (as opposed to binary data from all nodes). Our approach can be used to find a set of simultaneous anomalies affecting multiple connected subgraphs and the most likely sources of anomalies, whereas the above methods focus on finding only the most likely subset of one type of anomaly without finding the source(s) of that anomaly. The ability of our model to find the sources (root causes) of anomalies makes it more actionable and interpretable as well. Also, these other works cannot determine whether the network is in a healthy state, as they always find a subgraph with the highest anomaly score. Both Tango and Takahashi (2005) and Speakman et al. (2015) enforce a predefined neighborhood radius for which the prediction power may decrease when the size (depth) of the anomalies is larger than this radius. Also, their work is based on a user-defined score function that defines the anomalousness of a set of nodes satisfying a linear time subset scanning property. Because our model deals with probabilities rather than user-defined functions, we do not need to specify any anomaly detection model parameters (such as the neighborhood radius).

2.5. Summary of the Literature Review and Contributions

Sensor data in networks are complex because they are large in scale and heterogeneous, obtained from multiple sources, and have a high topological/spatial dependency. As a result, new network-specific models that can adapt to such complexity must be developed. We can conclude that although there are other network anomaly detection approaches that have their advantages and disadvantages, none of these approaches work well with all of the different graph structures, sensor data behavior, anomaly propagation scenarios, and modeling assumptions. The main limitation of existing approaches, which our paper intends to address, is their inefficiency in utilizing heterogeneous data, including node attributes (sensor data) and network typology collected from a subset of nodes across the network to detect multiple simultaneous anomalies and the most likely sources/causes of these anomalies. Also, models that focus only on independent nodes or a subgraph ranking are not effective because the detected nodes/subgraph may violate the dependencies embedded in the anomaly propagation paths or the connectivity requirement of anomalous nodes. By developing a framework that localizes anomalies and finds their sources by searching a subset of high-risk nodes and subnetworks that follow certain propagation paths, this article aims to provide more interpretable and actionable results. Compared with many available network anomaly detection models that require a specific distribution for sensor data or search parameters in their algorithms (e.g., neighborhood size), our model is more generic and can be applied in graph structures where no such information is available. Also, unlike many models that work only on tree structures, our framework can be applied to any network structure that can be represented by a directed acyclic graph, which covers a wide range of network systems, such as power and water distribution networks.

3. The Problem Setting for Network Monitoring

In this section, the structure of the proposed framework for network modeling and monitoring is formalized.

The list of the main notation in the paper is given in Table 1. The problem of anomaly detection can be formalized as follows:

Problem A (Multiple Anomaly Detection for Heterogeneous Networks). Given the topology of the attributed network, sensor locations and attributes, and anomaly propagation rules (paths) for each type of anomaly, aggregate/analyze real-time network sensor observations collected from sensors across the network to locate (a) the most likely sources and types of potential anomalies (if any) and (b) the impacted anomalous nodes/subgraphs.

The main assumptions made in this paper are the following. The system has a network structure made up of nodes or vertices connected by links or edges. The network is connected; its topology is represented by a known directed acyclic graph, which is a directed graph with no directed cycles. Also, both tree and nontree topologies can be considered for the topology of the network. The DAG representation of the network also helps with defining the anomaly propagation paths as defined in Section 3.3. The set of noisy binary sensor attributes (if any) collected at each node is known. Each sensor attribute may only partially reveal the state of the node where the sensor is located. One or multiple types of anomalies can occur simultaneously across the network. Because sensor attributes have different sampling rates, sensor time series are segmented according to a user-defined window size Δ, that is, time epoch t includes all time points between $[(t-1)\Delta, t\Delta)$. All model variables/parameters are defined based on this window size. Also, if the source of each anomaly is known, the propagation of anomalies across the network can be detected. We later relax this assumption by detecting anomalous subgraphs that contain only adjacent/connected subsets of nodes as an extension of our work.

3.1. Network Topology and Characteristic Variables

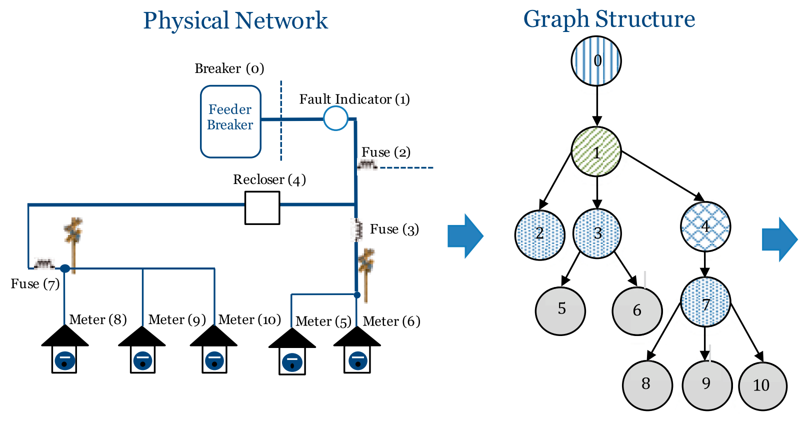
To monitor the health status of the network, or graph, a hierarchical structure is defined based on the physical topology of the network, consisting of nodes/vertices and edges/links. The network is assumed to have N distinct nodes where each node corresponds to a unique entity, such as a sensor, device, equipment, or user, and can be a potential candidate for the original location of anomalies. The adjacency matrix $\mathbf{A} = [a_{ij}]$, where $a_{ij} = 1$ if there is a direct link between node i and node j, is used to represent the topological structure of the network. There are several sensors in the network, which are placed at a selected set of nodes. For each node, the physical entities and sensors located at that node (if any) are known. We define a subnetwork or subgraph as a network formed by a subset of nodes and edges of the original network. The parameters to characterize a subnetwork are R_n denoting a subset of nodes with node n as the parent node and $S_{n'}$ denoting the set of sensor attributes generated at each node n' within subgraph R_n.

In Figure 1, a simple physical network with 11 nodes ($n = 11$) inspired by real distribution networks and the corresponding network graph structure are shown. The network has a breaker, a fault indicator, three fuses, one recloser, and five meters. Node i is the parent of the subgraph denoted by R_i. Also, node i can generate sensor attributes listed in S_i. For instance, for node 1, only attribute number 5 is available (i.e., $S_1 = \{5\}$). The connection between nodes is shown by an adjacency list, which is obtained from an adjacency matrix. For instance, for node 1, the adjacency list contains nodes 2, 3, and 4 representing three distinct edges from node 1 to nodes 2, 3, and 4. All information given in the network characteristic table is assumed to be known. For many network systems, such as electric grids,

Table 1. The List of the Main Notation Used in the Paper

Notation	Description
	Network topology
N	The number of nodes (vertices) in the network
$\mathbf{A} = [a_{ij}]_{N\times N}$	The adjacency matrix, where $a_{i,j} = 1$ if there is a direct link between node i and node j
R_n	The subgraph/subset of nodes with node n as the parent node (i.e., node n and all its downstream nodes)
U_n	The set of nodes upstream from node n
S_n	The set of attributes collected at node n
s_{nj}	A binary variable denoting whether sensor attribute j is collected at node n
	Anomalies and propagation
$O^*_m(t)$	The original location(s)/source(s) of the anomaly type m ($O^*_m(t) \in \{1,\ldots,N\}$) in epoch t
$x_{nm}(t)$	A binary variable indicating whether node n is under the effect of anomaly type m in epoch t
$o_{nm}(\mathrm{t})$	A binary indicator indicating whether the original location of the anomaly type m is at node n in epoch t
d_{ij}	A binary parameter denoting whether an anomaly propagates from node i to node j
	Sensor data
Δ	Observation interval
J	The number of binary sensor attributes in epoch t
$y_{nj}(t)$	The binary sensor output j observed from node n in epoch t ($y_{nj}(t) \in \{0,1,\emptyset\}$)

Figure 1. (Color online) An Example of a Small Segment of a Distribution Network with 11 Nodes (Node 0–Node 10) and Its Topological Characteristics



Network Characteristics

i	R_i	S_i	*Adjacency List*
0	{0,1,…,10}	{ }	[1]
1	{1,…,10}	{5}	[2,3,4]
2	{2}	{ }	[ ]
3	{3,5,6}	{ }	[5,6]
4	{4,7,8,9,10}	{4}	[7]
5	{5}	{1,2,3}	[]
6	{6}	{1, 2, 3}	[]
7	{7,8,9,10}	{ }	[8,9,10]
8	{8}	{1,2,3}	[]
9	{9}	{1,2,3}	[]
10	{10}	{1,2,3}	[]

Notes. Nodes of the same type are shown with similar patterns. Also, a fixed set of sensor attributes are collected from nodes of the same type.

the network topology may change to a new but known setting during operation for various reasons. Our framework can still be used for such systems because its only assumption is that the topology and the propagation sets are known at any decision epoch. In other words, the topology and propagation sets do not need to be fixed; however, they need to be known at any given time. We should point out that detecting changes in the topology of the network is beyond the scope of the paper.

3.2. Network Anomaly Variables

The network is subject to M different types of anomalies. Inspired by real networks, an anomaly can originate at a node and then continue to all nodes inside its anomaly propagation set. It is possible that multiple anomalies originate at multiple nodes. During any time interval of interest, each node in the network may either be impacted by an anomaly (state 1 to state M) or be under a normal condition (state 0). A generic state-space model is utilized to characterize the dependencies between each node's health status variables and the set of sensor data collected at each node. The vector $\boldsymbol{x}_n(t) = [x_{n1}(t), \ldots, x_{nM}(t)]$ is used to denote the anomaly status of node n in epoch t ($x_{nm}(t) \in \{0,1\}$). The full status of the network is known at epoch t if $\mathbf{X}(t)$ (that is, $\mathbf{X}(t) = \{\boldsymbol{x}_1(t), \ldots, \boldsymbol{x}_N(t)\}$) is known. To monitor the original sources of anomalies, we define a binary variable $o_{nm}(t)$ reflecting whether node n is the original location/source of the anomaly type m in epoch t. The network can be under the influence of many anomalies of the same type or different types at any time point. We define $O^*_m(t)$ to denote the original location(s)/source(s) of the anomaly type m ($O^*_m(t) \in \{1, \ldots, N\}$) in epoch t. It is clear that all these variables are unknown at any time epoch and need to be estimated. For each epoch t, the outcome of the proposed framework will be the estimated matrices $\mathbf{X}(t) = [x_{nm}(t)]$ and $\mathbf{O}(t) = [o_{nm}(t)]$.

3.3. Network Anomaly Propagation

To analyze anomalies in a structural manner while considering their propagation in the network, a binary deterministic belief network that models the propagation of anomalies across the network is employed according to the DAG representation of the network. The parameter d_{ij} is used to denote whether an anomaly propagates from node i to node j. Now the anomaly propagation paths across the network can be represented by a N by N matrix $\mathbf{D} = [d_{ij}]$. For mathematical convenience, we have assumed that the propagation between two nodes is the same for all anomaly types. Because of the sparse structure of large-scale networks, a sparse version of matrix $\mathbf{D}$ is utilized. The nodes along the propagation path of node n form the anomaly propagation set of node n. Also, all subgraph parameters (such as R_n) are determined based on the anomaly propagation set of node n. The relationship between anomaly location variable o and status variable x for each node according to its propagation can be defined as follows:

$$\begin{aligned}&o_{nm}(t) \le x_{n'm}(t),\\ &\quad n' \in R_n (\text{i.e.}, d_{nn'} = 1),\ \forall n \in \{1, \ldots, N\},\\ &\quad \forall m \in \{1, \ldots, M\}.\end{aligned} \tag{1}$$

For example, in a distribution network, the propagation of an outage in the network is fully known according to the location of protective devices. In such networks, all downstream nodes from the anomalous source are directly impacted. The anomaly propagation matrix in that case can be directly computed from the adjacency matrix as follows:

$$\mathbf{D} = \min\{[\mathbf{1}]_N, (\mathbf{A} + \mathbf{I}_N)^{\alpha}\},$$

where α is the depth of the graph representing the maximum length between two nodes, $\mathbf{I}_N$ is the identity matrix of size N, and $\mathbf{1}_N$ is an all-ones matrix. The integer variable α can be interpreted as the anomaly neighborhood radius by taking values smaller than the depth of the network. Because of the sparse structure of most large-scale networks, only positive $d_{i,j}$s may be recorded as a list variable. We relax the assumption of having predefined propagation paths in Section 5.6. In such a problem, the detected nodes do not need to follow predefined propagation rules. The only requirement is that the detected nodes are connected. Figure 2 shows how two independent anomalies originating at node 3 and node 4 are propagated in the network shown earlier in Figure 1. For the distribution network shown in Figures 1 and 2, the anomaly propagation list can be formed as follows:

$$\mathbf{D} = \{(0,1),\ldots,(0,10),(1,2),(1,3),(1,5),(1,6),\ldots,(3,5),\\ (3,6),(4,7),\ldots,(4,10),(7,8),(7,9),(7,10)\}.$$

3.4. Heterogeneous Network Sensor Data

There are J types of 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 sensor attributes). During any time interval of interest of length Δ $[(t-1)\Delta, t\Delta]$, node n is assumed to generate either the output $y_{nj}(t)$ (if a sensor that generates output j is at this node) or simply no output (if no sensor that generates output j is at this node), that is, $y_{nj}(t) = \emptyset$ for the measurement type j. In this work, we have focused only on binary attributes because of their wide application in many network structures, such as power distribution and communication networks. However, our model can handle nonbinary attributes as well. The causal relationship between network anomaly variables and sensor data can be shown by a directed acyclic graph as shown in Figure 3 for node n. Our framework can consider multiple observations within any decision interval by assuming that sensor data points are independent over time. For example, by assuming that there are W data points in interval $[(t-1)\Delta, t\Delta]$, sensor output $y_{nj}(t)$ can be computed as the summation of all W observations collected during that time interval. Our proposed framework cannot be used directly for other types of dynamic sensor data and temporal behavior.

Figure 2. (Color online) An Example of How Two Independent Anomalies Can Propagate in a Network

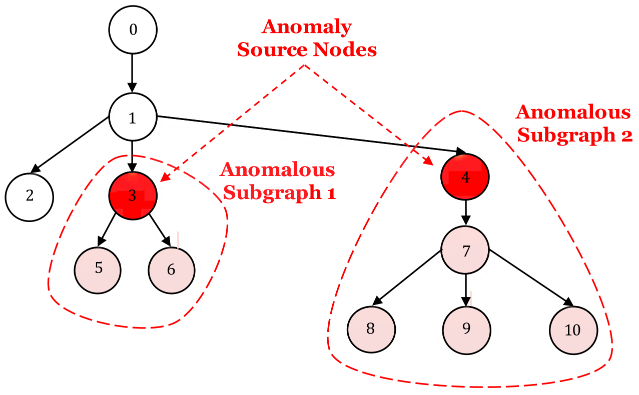
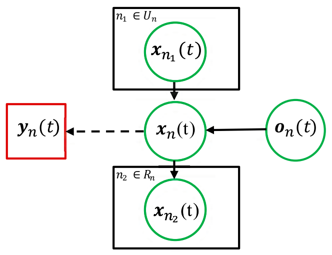


Figure 3. (Color online) A Graphical Model of a State-Space Model Representing the Dependencies Between Network Variables



Notes. The dashed edge is a stochastic edge, whereas the continuous edges are deterministic edges between two variables. A circle vertex represents a latent/hidden variable.

4. Node Anomaly Detection with Heterogeneous Network Data

In this section, we discuss the steps for network anomaly detection at the node level.

4.1. Network Topology and Data Aggregation

For notational convenience, we remove the time index in the rest of this section. As mentioned earlier, R_n is the subset of nodes with node n as the parent node. The idea of defining subgraph R_n is that if an anomaly occurs at node n, we know that all nodes in R_n should be affected. We can analyze the data and anomaly status of all nodes within each subgraph together to better diagnose the status of the nodes within that subgraph. For node n, the set of observed sensor attributes is denoted by $\boldsymbol{y}_n = [y_{n1}, \ldots, y_{nJ}]$. Depending on the type of node n, one or more elements of $\boldsymbol{y}_n$ may not exist. In addition to sensor attributes observed at node n, there is a set of measurements from all neighbor sensors $\boldsymbol{y}_{n'}$ where $n' \in R_n$ in the subgraph where n is the parent node. It is obvious that for subgraphs across the network, the feature inputs to detect anomalies are not the same; thus, network data are heterogeneous. To accommodate both the topology and available sensor signals within each subgraph, we propose to first convert the topology and set of signal outputs within each subgraph to two distinct sets of data vectors: (a) aggregated sensor data, denoted by $\bar{\boldsymbol{y}}_n$, and (b) aggregated sensor topology data, denoted by $\bar{\boldsymbol{s}}_n$, for all nodes in the network. The aggregated sensor data are the set of data points that summarizes the observations from sensor attributes within the subgraph. The aggregated sensor topology data summarize the number and types of sensors within the subgraph. For the *j*th sensor attribute, the aggregated vector can be represented by a two-tuple multiset, where the first element is the sum of the original binary sensor observations and the second element is the number of sensors that can generate signal *j* in the subgraph. To have a normalized representative vector, we can divide the aggregated values by the number of available sensors. In summary, the process of aggregation for (a) and (b) is conducted through predefined aggregation functions around the effective neighborhood of each node as follows:

$$\bar{y}_{nj} = \left[\sum_{n' \in R_n} y_{n'j}\right]\left[\sum_{n' \in R_n} \mathbb{1}\{s_{n'j} = 1\}\right]^{-1}, \quad \forall n \in \{1,\dots,N\},\ \forall j \in \{1,\dots,J\}, \tag{2}$$

$$\bar{s}_{nj} = \left[\sum_{n' \in R_n} \mathbb{1}\{s_{n'j} = 1\}\right]\left[\sum_{n' \in \{1,\dots,N\}} \mathbb{1}\{s_{n'j} = 1\}\right]^{-1}, \quad \forall n \in \{1,\dots,N\},\ \forall j \in \{1,\dots,J\}. \tag{3}$$

The type of aggregation function selected is important because it determines how much information is lost during the aggregation process. For instance, one may use the median instead of the mean because of its robustness against outliers. Now, for subgraph R_n, the network sensor data can be represented by a $2J$-dimensional vector $[\bar{y}_{n1},\dots,\bar{y}_{nJ},\bar{s}_{n1},\dots,\bar{s}_{nJ}]$. If signal *j* is not collected in subgraph *n*, then both $\bar{y}_{nj}$ and $\bar{s}_{nj}$ are set to zero. By having both variables, we can simply distinguish between actual values of zeros and zeros due to no sensors in the subgraph. It can be observed from Equations (2) and (3) that all elements in the aggregated vectors are normalized between zero and one. The core idea here is that nodes and subgraphs with similar types of sensors get a similar cluster vector. Also, different subnetworks with any topology, types of sensors, and observed sensor attributes can be represented by a standard vector that has the same size across all subnetworks. To illustrate the concept of data aggregation, we provide a simple example in Figure 4 for the distribution network given earlier in Figure 1. The sensor attribute vectors $\boldsymbol{y}_1,\dots,\boldsymbol{y}_{10}$ as well as the aggregated vectors for two subgraphs R_1 and R_4 are shown in the figure. It can be seen that for both subnetworks, the topology and sensor attributes are transformed into two five-dimensional standardized vectors $\bar{\boldsymbol{y}}_1$ and $\bar{\boldsymbol{s}}_1$ for R_1 and $\bar{\boldsymbol{y}}_4$ and $\bar{\boldsymbol{s}}_4$ for R_4.

4.1.1. Statistical Properties of the Aggregation Function.

As shown earlier, we chose a scalable aggregation function that provides a standardized vector representation of any subgraph with any set of sensors. We will show that this aggregation function is injective and statistically sufficient (under certain conditions) with regard to detecting anomalies and captures all the information concerning the subgraph that is relevant for anomaly detection. The aggregation functions should be defined so that as much information as possible is maintained on both topology and sensor attributes. It is known that aggregation functions should be injective to attain maximum discrimination power (Seo et al. 2019). In Remark 1, we discuss the injectivity of the selected aggregation functions, which indicates how the aggregation functions map two different subgraphs to two different representation vectors.

Remark 1 (Injectivity of the Aggregation Functions). For any two subgraphs R_{n_1} and R_{n_2} ($n_1, n_2 \in \{1,\dots,N\}$), the aggregated representation vectors (which are standardized between zero and one) are the same if and only if the number of sensors generating the same types of attributes is the same in both subgraphs and the cumulative observations for all sensor attributes are the same as well.

Based on Remark 1, two distinct subgraphs with different sets of sensors and/or cumulative sensor observations are represented distinctively after being aggregated by Equations (2)–(3). In Remark 2, we show that the sum operators over sensor data and sensor numbers incur statistically sufficient measures with regard to the anomaly status of any subgraph for each sensor attributes.

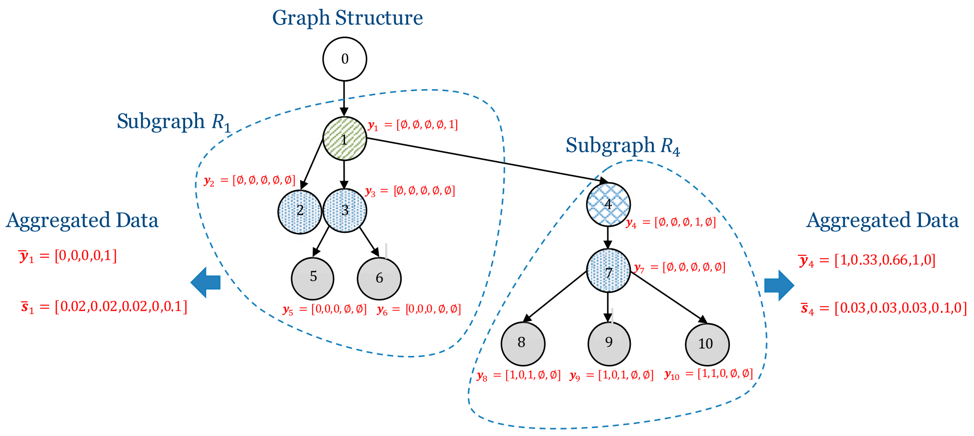
Remark 2 (Statistical Sufficiency of the Aggregation Functions for Each Sensor Attribute). For any subgraphs R_n the following holds true for each sensor attribute *j* under certain conditions:

$$\Pr(x_{nm} = 1 \mid y_{n'j}; n' \in R_n) \propto \Pr\left(x_{nm} = 1 \,\middle|\, \sum_{n' \in R_n} y_{n'j}, \sum_{n' \in R_n} \mathbb{1}\{s_{n'j} = 1\}\right) \quad \forall j \in \{1,\dots,J\},$$

that is, $\sum_{n' \in R_n} y_{n'j}$ and $\sum_{n' \in R_n} \mathbb{1}\{s_{n'j} = 1\}$ are sufficient to infer the anomaly probability distribution of the parent node of subgraph R_n based on data collected for sensor attributes *j* from all nodes in that subgraph. The proof is given in Appendix A in the online supplement.

Remark 2 is valid for categorical features, too (beyond the scope of this paper). However, it does not apply to continuous attributes or when the conditional independence assumption is not met. For such

Figure 4. (Color online) A Simple Example of a Physical Network (11 Nodes) and Aggregating Network Structure and Sensor Data


Graph Structure
0
Subgraph R_1
1
$y_1 = [\emptyset, \emptyset, \emptyset, \emptyset, 1]$
$y_2 = [\emptyset, \emptyset, \emptyset, \emptyset, \emptyset]$
2
3
$y_3 = [\emptyset, \emptyset, \emptyset, \emptyset, \emptyset]$
5
6
$y_5 = [0,0,0, \emptyset, \emptyset]$
$y_6 = [0,0,0, \emptyset, \emptyset]$
Aggregated Data
$\bar{y}_1 = [0,0,0,0,1]$
$\bar{s}_1 = [0.02,0.02,0.02,0,0.1]$
Subgraph R_4
4
$y_4 = [\emptyset, \emptyset, \emptyset, 1, \emptyset]$
7
$y_7 = [\emptyset, \emptyset, \emptyset, \emptyset, \emptyset]$
8
9
10
$y_8 = [1,0,1, \emptyset, \emptyset]$
$y_9 = [1,0,1, \emptyset, \emptyset]$
$y_{10} = [1,1,0, \emptyset, \emptyset]$
Aggregated Data
$\bar{y}_4 = [1,0.33,0.66,1,0]$
$\bar{s}_4 = [0.03,0.03,0.03,0.1,0]$

Notes. Nodes of the same type are shown with similar patterns. The aggregated variables are calculated given that there are a total of 100 meters, 10 fault indicators, and 10 reclosers.

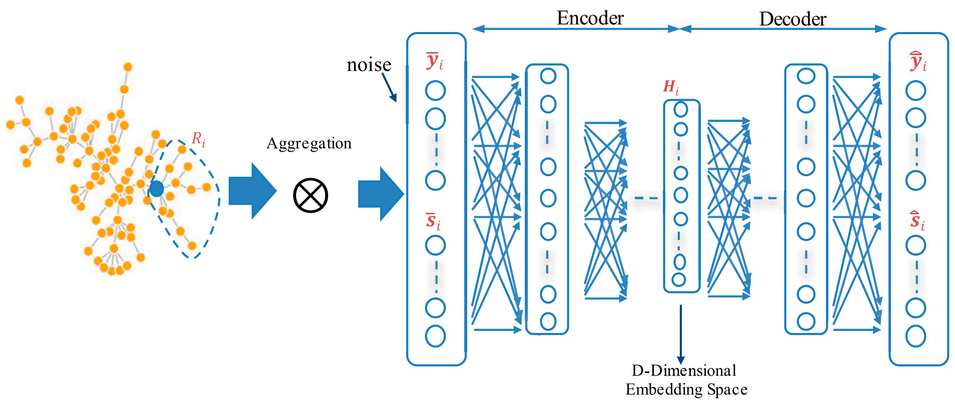
cases, we need to choose or design aggregation functions so that minimum information (from sensor data and sensor topology) is lost. We should note that the aggregation functions do not need to follow the properties discussed in Remarks 1 and 2. These remarks only motivated us to employ the aggregation functions given in Equations (2)–(3). One may include more elements of the topology/sensor data.

4.2. Optional Dimensionality Reduction with Autoencoder Embedding

After aggregating the topology and sensor data within each subgraph, the output vectors may become high dimensional, depending on the aggregation functions used and the number of possible sensors and sensor attributes. The objective here is to utilize the information contained in the aggregated sensor data and aggregated sensor topology data and map each subgraph into a D-dimensional vector in an entirely unsupervised manner. The transformation should be designed in a way that it can be applied to any subgraph of the network regardless of its size or sensor topology. The embedding problem is conducted through an autoencoder, which is a feedforward neural network with an input layer, an output layer, and one or more hidden layers. The idea is to use the representation learned from the autoencoder in the low-dimensional bottleneck layer as the input for the node anomaly detection task. The autoencoder has two main mapping functions: encoder and decoder. The encoder transforms the aggregated node data in the input layer to a low-dimensional representation (latent embedding space). Then, the decoder returns a reconstruction of the original inputs from the low-dimensional embedding space. The loss function is the minimization of the difference between node inputs and their reconstruction outputs. The main idea behind this autoencoder is that if the decoder can return a reasonable reconstruction of the original inputs from the low-dimensional embedding space, we can transform aggregated network data at each subgraph into a lower dimensional space and use that as an input for anomaly detection. The reconstruction error can be used on a test set to determine whether this optional step is useful and should be used, depending on the complexity and dimension of the sensor attributes. After the training is performed on the autoencoder, we can disregard the decoder for the task of anomaly detection. The number of hidden layers and the number of neurons in each layer are fine-tuned with cross-validation.

An overview of the network data compression process is shown in Figure 5. The hidden layer in the middle with D neurons (i.e., the smallest number of neurons among layers) is the one to be used as a new feature vector for anomaly detection. Each subgraph and its sensor data can be transformed into an aggregated and compressed D-dimensional space, which is designed for minimum information loss and controlled by the autoencoder hyperparameters. To improve the robustness of the network and obtain better generalization and faster learning, we can add noise to the aggregated sensor data ($\bar{y}$) as the input of the autoencoder while keeping the network structure fixed. In addition to the potential to compress large-scale network structures and sensor data and detect redundant/correlated attributes, the autoencoder can accommodate nonlinearity, which is an advantage over linear fusion models, such as Principal Component Analysis. We should point out that we did not directly employ the embedding of the nodes for all subgraphs for the following reasons. Available models in the literature for embedding an attributed graph require the

Figure 5. (Color online) Overview of a Deep Autoencoder



Note. The output is the encoder section that can transform each subnetwork aggregated topology and sensor data into a D-dimensional embedding space.

same graph topology and sensor output dimensions for all nodes. However, it is assumed in our work that the topological structure (e.g., size, type of sensors, and sensor outputs) of subgraphs differs from each other. Thus, we cannot use a single embedding function to transform all nodes into an embedding space. Embedding after the aggregation step can significantly lower the complexity of the problem without losing important information concerning the subgraph that is relevant for anomaly detection. This step is entirely optional, and that is why we separated it from the aggregation step. We should point out that there is no theoretical guarantee that this entirely optional step will improve the anomaly detection results. We will only numerically evaluate its performance in Section 6.

4.3. A Multilayer Perceptron for Nodal Anomaly Detection and Ranking

Consider a random decision epoch in which sensor observations from all sensors across the network are collected. The problem of anomaly detection defined earlier is divided into Problem A.1 and Problem A.2. For any subnetwork, Problem A.1 is formalized as follows:

Problem A.1 (Independent Anomaly Detection for Network Nodes). Given the topology of subgraph R_n and the set of sensor observations for all sensors within the subgraph, (a) what is the probability that subnetwork R_n is under the impact of anomaly type m, denoted by γ_{nm}, and (b) what is the most likely status of node n, denoted by v_n, where $m \in \{0, 1, \ldots, M\}$ and $m = 0$ refers to the no anomaly condition?

To address Problem A.1, we can build a multilabel anomaly classifier and train it with past data. For any subnetwork, we can first transform the aggregated vectors of the corresponding network structure and sensor data into a D-dimensional embedding space (optional). The new D-dimensional vectors can then be used as the inputs of the multiclass anomaly detection classifier. Such a classifier is trained to predict whether the entire subnetwork is under anomaly and, if yes, predict the most likely anomaly type. The difference between this approach and a multilabel classifier that uses direct sensor attributes from each node is that here the topology of the subnetwork and the entire set of sensor data within the subgraph are used. Also, the prediction will cover the entire subnetwork as opposed to diagnosing one node at a time. This is different from working only on individual nodes and not the anomaly neighborhood where an anomaly can propagate. In what follows, the relationship between M types of anomalies and transformed signals for each subnetwork in the embedding space is modeled through a deep neural network. We first encode the anomaly class labels via one-hot encoding, where each element of the output vector corresponds to an anomaly type. We consider the first class to be class 0, in which the subnetwork is not affected by any anomaly. A multilabel classifier inspired by multilayer perceptron networks (Gulli and Pal 2017) is proposed here. The corresponding MLP has four main layers: (i) the input later $\mathbf{H}$ with D heterogeneous neurons representing the aggregated subgraph data in the embedding space, (ii) the output layer $\mathbf{X}$ with $M + 1$ binary neurons representing M anomalies and one normal condition, (iii) hidden layer 1 with H_1 hidden neurons representing subgraph latent factors (denoted by $\{z_1^{\{1\}}, \ldots, z_{H_1}^{\{1\}}\}$), and (iv) hidden layer 2 with H_2 hidden neurons representing anomaly latent factors (denoted by $\{z_1^{\{2\}}, \ldots, z_{H_2}^{\{2\}}\}$). The latent variables can account for the fact that *anomalies and network data might come from a smaller or larger number of latent factors*. For instance, there might be several anomalies that exhibit similar characteristics and are similar for generating a set of highly related sensor attributes. These latent factors allow the model to incorporate marginal dependencies and interactions within anomalies and sensor attributes, which could not be achieved with approaches that work only under the independence assumption of anomalies and sensor attributes. The hyperparameters H_1 and H_2 can be tuned in by cross-validation. The high-level connection between the elements in each layer is shown in Figure 6. In this figure, $\boldsymbol{b}^{\{1\}} = [b^{\{1\}}_{1:H_1}]$, $\boldsymbol{b}^{\{2\}} = [b^{\{2\}}_{1:H_2}]$, and $\boldsymbol{b}^{\{M\}} = b^{\{X\}}_{1:M+1}$ are the bias variables in hidden layers 1 and 2 and output layer $\mathbf{X}$ that together control the uncertainty. The elements in $\mathbf{B}^{\{1\}} = [\beta^{\{1\}}_{d,h_1}]_{D\times H_1}$, $\mathbf{B}^{\{2\}} = [\beta^{\{2\}}_{h_1,h_2}]_{H_1\times H_2}$, and $\mathbf{B}^{\{X\}} = [\beta^{\{X\}}_{h_2,m}]_{H_2\times M+1}$ quantify the contribution of variables between layers $\mathbf{H}, 1, 2,$ and $\mathbf{X}$.

Note that if we multiply $\mathbf{B}^{\{1\}}$, $\mathbf{B}^{\{2\}}$, and $\mathbf{B}^{\{X\}}$, we get a $D \times (M+1)$ matrix, where its dmth element can be interpreted as the contribution of anomaly type m on the dth network embedding variables (or vice versa). The equations for each layer are defined below:

$$z^{\{1\}}_{h_1} = \omega^{\{1\}}\left(\sum_{d=1}^{D} \beta^{\{1\}}_{dh_1} h_{nd} + b^{\{1\}}_{h_1}\right), h_1 \in \{1, \ldots, H_1\}, \tag{4}$$

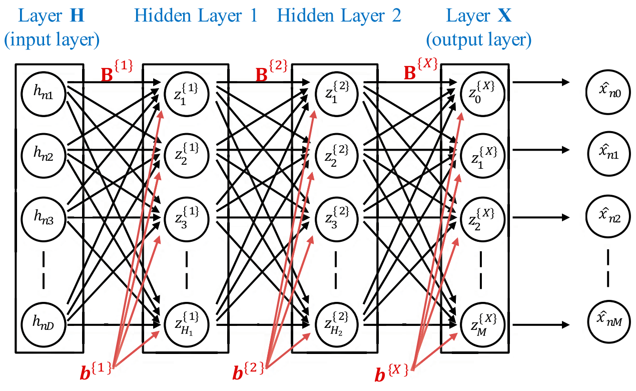
$$z^{\{2\}}_{h_2} = \omega^{\{2\}}\left(\sum_{h_1=1}^{H_1} \beta^{\{2\}}_{h_1 h_2} z^{\{1\}}_{h_1} + b^{\{2\}}_{h_2}\right), h_2 \in \{1, \ldots, H_2\}, \tag{5}$$

$$z^{\{Y\}}_{m} = \sum_{h_2=1}^{H_2} \beta^{\{X\}}_{h_2 m} z^{\{2\}}_{h_2} + b^{\{X\}}_{m}, m = \{0, \ldots, M\}, \tag{6}$$

where $\omega^{\{1\}}$ and $\omega^{\{2\}}$ are parameter-free activation functions. Now to generate the output as a unit probability vector (that is, a normalized probability distribution consisting of $M + 1$ probabilities), we can use the softmax function (multinomial logistic regression) as follows:

$$\Pr(x_{nm} = 1 \mid \boldsymbol{h}_n) = e^{z^{\{X\}}_m} \left[\sum_{m=0}^{M} e^{z^{\{X\}}_m}\right]^{-1}, m = \{0, \ldots, M\}, \tag{7}$$

Figure 6. (Color online) The Structure of the MLP with Two Hidden Layers (Inputs Are Shown Based on Node n)



where vector $\boldsymbol{h}_n$ is the embedded vector subnetwork R_n. One aspect of the above formulation is its ability to be rewritten in a compact matrix form that helps calculate the recursive equations needed for efficient learning and inference. For a set of input-output points stored in $\mathcal{D} = [\mathbf{H}, \mathbf{X}]$ that includes $|\mathcal{D}|$ row-vector data points, we have

$$\mathbf{Z}^{\{1\}} = \Omega^{\{1\}}(\mathbf{H}\,\mathbf{B}^{\{1\}} + \mathbf{b}^{\{1\}}\mathbf{I}), \mathbf{Z}^{\{2\}} = \Omega^{\{2\}}(\mathbf{B}^{\{2\}}\mathbf{Z}^{\{1\}} + \mathbf{b}^{\{2\}}\mathbf{I}), \mathbf{Z}^{\{Y\}} = \mathbf{B}^{\{X\}}\mathbf{Z}^{\{2\}}, \tag{8}$$

where $\Omega^{\{:\}}$ is the element-wise activation function for a matrix. It can be observed that the only unknown parameters of this model are $\boldsymbol{\Theta} = \{\mathbf{B}^{\{:\}}, \boldsymbol{b}^{\{:\}}\}$. Note that these parameters are shared between all subgraphs with any size and any set of sensor data. To use the model in real time, we need to train the MLP structure. Let us assume that multiple samples of network data are available for all the N nodes in the network. The training data set can be shown by $\mathcal{D} = \{\mathbf{H}, \mathbf{X}\}$, which includes transformed network data from the autoencoder ($\mathbf{H}$) and the true anomaly status of all subnetworks ($\mathbf{X}$) for different time epochs. Let us denote x_{nm} as the true anomaly status of node n (and subnetwork R_n) with respect to anomaly type m and $\hat{x}_{nm}$ as its estimated output from the MLP. The categorical cross-entropy function is used as the loss function in the back-propagation algorithm. With this function, the MLP will be trained to output a probability over the $M + 1$ anomaly classes for each node. The back-propagation procedure will follow the standard feed-forward and back-propagation steps. In our experiments, we tune all hyperparameters through a grid search.

4.3.1. Overview of the MLP.

Although this paper does not contribute to deep learning and the theoretical foundations of MLP, it customizes MLP as a highly flexible tool that has appealing properties because of the existence of structural latent factors, the variety of sensor attributes, and the potential for deep learning and inference. In this structure, all anomalies and sensor attributes are incorporated in a unified model, which can account for the fact that many anomaly events and sensor attributes are highly correlated and should not be analyzed independently. Also, the topology of the network and its sensors are taken into account. With defining structural latent factors, the overall interpretability is improved; marginal dependencies between anomalies and sensor attributes and network topology are allowed. With defining latent factors, the number of model parameters can become lower than DM in high-dimensional settings with large M and large J, contributing to *lowering computational complexity*. A large set of latent factors may be defined to capture more *interaction* and *complexity* between anomalies, sensor attributes, and network topology. Although it requires more computational time for large data sets and a large number of anomalies, considering a single network for an M-class anomaly detection task can theoretically result in optimal classification when enough data are available. The MLP structure has a generic structure that accommodates well-known parametric settings. By setting $\mathbf{B}^{\{1\}}$ and $\mathbf{B}^{\{2\}}$ to identity matrices of size D and M, respectively, and setting ω to an identity function, we can reach multinomial logistic regression.

4.4. Summary of the Proposed Approach for Node Anomaly Detection (Model $\mathcal{M}_0$)

An overview of the proposed approach for node/subgraph anomaly detection, referred to as the baseline model or $\mathcal{M}_0$, is given below. For training, we need to first preprocess network data using the aggregation functions and the autoencoder network. The inputs of the autoencoder are the series of aggregated subgraph data without their labels. Once the autoencoder network is trained, the aggregated network data at the subgraph are transformed into the embedding space. The transformed embedded data together with the labels of the nodes are used to train the MLP in Figure 6. Once the MLP is trained, we can use it for any new sample r to predict the probability of being under each anomaly condition (i.e., $\Pr(x_{rm}=1)$) and find the most probable anomaly class v_n as

$$\mathcal{M}_0 : v_n = \arg \max_{m\in\{0,\ldots,M\}} \Pr(x_{rm}=1\,|\,\boldsymbol{h}_n). \tag{9}$$

We can also use the probability outcome of the MLP to rank the nodes and subnetworks with respect to the risk of being under an anomaly. Algorithm 1 in Appendix B of the online supplement summarizes the details of the entire process to transform raw network data into detection results at the node level.

5. The Proposed Network-Based Anomaly Detection Framework

The results obtained from the framework in Section 4 only provide the anomaly status of each node independent of other nodes in the network. Thus, the framework ignores the data collected from other sensors and the status of other nodes outside the corresponding subgraph. In other words, it does not provide integrated insight regarding the entire network. The result from the node-to-node anomaly detection cannot locate the original location or identify the type of anomalies. However, it only shows whether a node and its subgraph are under an anomaly condition. The nodal approach also cannot verify whether the network is under multiple simultaneous anomalies. With respect to the status of the entire network, the results from the nodal approach may not be consistent with each other. For instance, for a parent-child case, nodal analysis may predict the parent node to be anomalous but the child node to be normal, which violates the anomaly propagation rule for the parent node. The objective of the anomaly detection framework here is to efficiently employ the results from nodal analysis across the network to (i) detect whether there are anomalies in the network and (ii) identify the sources of anomalies and all impacted nodes. Two optimization models, referred to as Model $\mathcal{M}_1$ and Model $\mathcal{M}_2$, are designed where the results from the multiclass MLP (i.e., the probability of being under each type of anomaly and the most likely status of each node) are utilized as inputs. The outcome is the most likely status of the entire network. Such results would divide the network into multiple distinct subgraphs/segments, each with a uniform anomaly status. This problem is formalized as follows:

Problem A.2 (Anomaly Detection for the Entire Network). Given the results from Problem A.1, namely, γ_{nm} and v_n, for all $n\in\{1,\ldots,N\}$, and $m\in\{0,1,\ldots,M\}$, (a) detect whether there are anomalies in the network and (b) identify the sources of anomalies and all impacted nodes.

We discuss important optimal properties of these models and propose solutions to reduce their computational complexity. The main difference between these two optimization models is the manner in which they utilize the outcome of the node-level anomaly detection phase (MLP). Model $\mathcal{M}_1$ employs the probability distribution of an anomaly for each node, whereas $\mathcal{M}_2$ employs each node's predicted label.

5.1. Anomaly Detection with Stochastic Nodal Outputs from MLP ($\mathcal{M}_1$)

As discussed in Section 4, the original outcome of the MLP for any node at any time point is the probability distribution of its anomaly status, which can be denoted by the vector $\boldsymbol{\gamma}_n(t)=[\gamma_{nm}(t)]$, where γ_{nm} is the probability of node n being under anomaly type m for $m\in\{0,\ldots,M\}$. Below, we develop an optimization model that can find the most likely locations of anomalies across the network. The outcome of this optimization model can divide the network into Q ($Q\geq 1$) mutually exclusive and distinct segments. Each segment contains a subgraph where all the subgraph nodes are under the same anomaly condition. We define $o_{nm}(t)$ as a binary decision variable reflecting whether node n is the source of anomaly type m. When $o_{nm}(t)=1$, then we have a subgraph with node n as a parent node that hosts the original location of an anomaly of type m, and all nodes in R_n are its impacted children. Also, when node n is not the source of anomaly m, it can either be under no anomaly or under an anomaly originating from an upstream node. The relationship between network nodes and anomaly location variables (denoted by o) and status variables (denoted by x), anomaly propagation paths (represented by edges), and sensor data can be represented by a directed acyclic graph with a deterministic causal relationship between the anomaly location and status variables and probabilistic dependencies between any node's status variable x and sensor data collected at that node. Before the first optimization model is introduced, the following two important remarks are stated.

Remark 3. Given the assumption that each node can only be under the impact of one anomaly at a time, the maximum number of anomalies within each subnetwork cannot exceed the number of nodes at the lowest level (leaf nodes) of that subnetwork (denoted by l_n for node n). Here leaf nodes are the ones that have no other nodes in their propagation paths.

Remark 4. Without taking into account the dependencies between network nodes from the propagation of anomalies, given the outputs of the MLP as $\Gamma = [\gamma_{nm}]_{N\times(M+1)}$, the most likely state for network variables in $\mathbf{O}(t) = [o_{nm}(t)]$ and subsequently all variables in $\mathbf{O}(t) = [o_{nm}(t)]$ is the one that maximizes the following posterior probability/belief:

$$\Pr(\mathbf{O}(t) \mid \boldsymbol{h}_1, \ldots, \boldsymbol{h}_n) = \prod_{n=1}^{N}\prod_{m=1}^{M}\gamma_{nm}(t)^{o_{nm}(t)} \times \prod_{n=1}^{N}\gamma_{n0}^{\left[1-\sum_{n'=1}^{N}\sum_{m=1}^{M}o_{n'm}(t)d_{n'n}\right]}, \tag{10}$$

where $\left[1-\sum_{n'=1}^{N}\sum_{m=1}^{M}o_{n'm}(t)d_{n'n}\right]$ is a binary term equal to one if node n is under the impact of no anomaly. Instead of $\gamma_{nm}(t)$, we can use the joint anomaly probability of all nodes in the propagation path of node n (i.e., $\prod_{n\in R_n}\gamma_{nm}(t)$) and get an alternative anomaly score for the network. Remark 3 is true because any number of anomalies greater than the number of leaf nodes, which are the nodes with no child nodes/downstream connections, results in at least one node being under the impact of two anomalies. Equation (10) in Remark 4 is the joint probability of all nodes' health status variables inspired by the chain rule in Bayesian belief networks (the proof is omitted because of its simplicity). By taking the log of the belief function in Equation (10) as the objective function, we can build an optimization model that takes into account the topological dependencies between nodes as structural constraints. The network optimization problem for anomaly detection can be formalized through the following binary integer programming problem:

$\mathcal{M}_1$: Optimal Network Segmentation Based on Anomalies−Model 1

$$\max_{o_{nm}(t),\forall n,m} z = \sum_{n=1}^{N}\sum_{m=1}^{M}o_{nm}(t)\log\gamma_{nm}(t) + \sum_{n=1}^{N}\left[1-\sum_{n'=1}^{N}\sum_{m=1}^{M}o_{n'm}(t)d_{n'n}\right]\times\log\gamma_{n0}(t), \tag{11}$$

$$\text{Subject To}: \quad \sum_{m=1}^{M}o_{nm}(t) \le 1, \quad \forall n \in \{1,\ldots,N\}, \tag{12}$$

$$\sum_{m=1}^{M}\left[\sum_{n'=1,n'\ne n}^{N}o_{n'm}(t)d_{nn'}\right] \le l_n\left(1-\sum_{m=1}^{M}o_{nm}(t)\right), \quad \forall n \in \{1,\ldots,N\}, \tag{13}$$

$$o_{nm}(t) \in \{0,1\}, \ \forall n \in \{1,\ldots,N\}, \ \forall m \in \{0,\ldots,M\}. \tag{14}$$

The objective function is defined according to the propagation of anomalies in the network. The left part of the objective function is only activated for anomalous nodes, whereas the right part is activated for all nodes that are not part of the anomaly propagation paths of the anomalous source nodes. The first constraint ensures that each node can be the source of only one anomaly at a time (note that the network can still be under simultaneous anomalies). The second constraint has multiple roles regarding the propagation of anomalies. First, it guarantees that no node along the propagation set of node n is the source of any anomaly if node n is the source of an anomaly. Also, it ensures that node n cannot be the source of any anomaly if one of the nodes within its anomaly propagation set is the source of the anomaly. The constant l_n shows the maximum number of anomalies within subgraph R_n. Because Equation (1) holds for the relationship between variables o and x, the optimal solution of $\mathcal{M}_1$ automatically provides all the optimal values for the elements of x. In Section 5.4, we provide some important remarks that help lower the complexity of this optimization model for large networks.

5.2. Anomaly Detection with Deterministic Nodal Outputs from MLP ($\mathcal{M}_2$)

Instead of using the outcome of the MLP in a probabilistic manner as in $\mathcal{M}_1$, we can alternatively use the predicted label of each node. Such a predicted label for node n, denoted by $v_n(t)$, can be estimated from Equation (9). Now, we have a network where all nodes are labeled as $[v_1(t), v_2(t), \ldots, v_N(t)]$. The objective of this network anomaly detection approach is to find the most likely set of distinct anomaly subgraphs and affected nodes given the predicted label of all nodes, which provides a segmented network with detected anomalies and impacted nodes. In what follows, we use the power of the MLP model to detect each node's anomaly status and use it to determine the status of all nodes and of the entire network. After MLP is trained, we can calculate the probabilistic confusion matrix $\mathbf{P} = [p_{i,j}]$, where $p_{i,j}$ indicates the average probability that a sample with true anomaly type i is classified into anomaly type j. Now for node n with the predicted anomaly $v_n(t)$, the probability of the true label being i equals $p_{i,v_n(t)}$. Thus, the probability of node n being the source of anomaly type m depends on the predicted probability values of all nodes inside the anomaly propagation set of node n (that is, all nodes in R_n) and the values in its probabilistic confusion matrix. Using p instead of γ, a binary optimization problem to find variables $o_{nm}(t)$ can be built as follows:

$\mathcal{M}_2$: **Optimal Network Segmentation Based on Anomalies−Model 2 :**

$$\max_{o_{nm}(t),\forall n,m} \quad z = \sum_{n=1}^{N}\sum_{m=1}^{M} o_{nm}(t)\log p_{m,v_n(t)} + \sum_{n=1}^{N}\left[1-\sum_{n'=1}^{N}\sum_{m=1}^{M} o_{nm}(t)d_{n'n}\right]\times \log p_{0,v_n(t)}$$

$$\text{Subject to :} \qquad \text{Equations (12)–(14).}$$

The left side of the objective function is activated if node n is found to be the location of an anomaly. The right side of the objective function is activated when node n and none of its connected ancestors are under the effect of an anomaly. It is clear that the complexity of Models 1 and 2 is the same, as they have the same number and types of constraints and variables.

5.3. Remarks for the Number of Network Segments

Below, a few remarks are introduced regarding the complexity of the proposed optimization problems.

Remark 5. The maximum possible number of mutually disjointed anomalous segments (denoted by Q^*) in a network with N nodes and known $R_1,\ldots,R_n$ can be found from the solution of the following binary integer programming, which has a similar structure to the maximum set packing problem:

$$\text{maximize} \quad Q = \sum_{n=1}^{N} q_n, \tag{15}$$

$$\text{Subject to :} \quad \sum_{n=1}^{N} \mathbb{1}\{s\in R_n\}\, q_n \le 1, \quad \forall s\in\{1,\ldots,N\}$$
$$q_n \in \{0,1\}, \qquad \forall n\in\{1,\ldots,N\}. \tag{16}$$

The value of Q^* represents the vulnerability of a network and helps with the efficient allocation of monitoring and restoration resources. The proof of this remark is in Appendix C in the online supplement.

Remark 6. Regarding the optimal solution of $\mathcal{M}_1$ and $\mathcal{M}_2$, if we consider all nodes with no anomaly as one segment, then the total number of optimal distinct segments at time t is

$$\sum_n\sum_m o^*_{nm}(t) + \mathbb{1}\left\{\sum_n\sum_m o^*_{nm}(t) < N\right\}.$$

This is an important remark for network operators because it gives the number of anomalous segments for the network at any time point. The proof for this remark is very simple because every nonzero $o_{nm}(t)$ refers to an anomalous subgraph with anomalous nodes of the same type. Because two subgraphs with anomalies cannot overlap each other, there is a distinct segment for each nonzero $o_{nm}(t)$. If the total number of impacted nodes is less than the total number of nodes, then there is at least one node under normal conditions. Because all normal nodes are considered as one segment, then the total number of segments will be added by one. This completes the proof.

5.4. Computational Complexity Remarks for Large-Scale Networks

A major challenge in analyzing large-scale network data are scalability. Four important remarks are introduced regarding the complexity of the proposed optimization problems. These remarks are effective for both Model 1 and Model 2 and will be used as initialization (Preoptimization) steps, which aim to lower the number of decision variables/constraints in the corresponding optimization problems.

Remark 7. For node n, if the optimal value of $\sum_m o^*_{nm}(t)$ is one, then the optimal m in o^*_{nm} equals

$$\arg\max_m \gamma_{nm}(t) \text{ in Model 1 and}$$
$$\arg\max_m p_{m,v_n(t)} \text{ in Model 2.}$$

Based on this remark, we can lower the number of variables needed in both models from $N\times M$ to N by keeping only $o_{nm^*}(t)$ instead of $[o_{n1}(t),\ldots,o_{nM}(t)]$. This is a significant computational advantage, particularly for large networks and networks with many types of anomalies.

Proof. From Equation (11) in Model 1, we can see that if $\sum_m o^*_{nm}(t)$ is one (i.e., node n is the location of an anomaly), then the right-hand side of the objective function is inactive. Given that the constraint coefficients for variables $o_{nm}(t)$ are the same for all m, then $\arg\max_m \gamma_{nm}(t)$ would make the objective function optimal. This completes the proof. The same proof applies to Model 2.

Remark 8. For any node n, if

$$\prod_{n'\in R_n} \gamma_{n'0}(t) > \max_m \gamma_{nm}(t)$$

$$\text{for Model 1 and } \prod_{n'\in R_n} p_{0,v_{n'}(t)} > \max_m \; p_{m,v_n(t)}$$

$$\text{for Model 2,}$$

then $o^*_{nm}(t) = 0$ for all $m \in \{0,\dots,M\}$, that is, node n is not the source of any anomaly.

Proof. This is an important finding because if the above condition is met, then we can remove all M decision variables for node n from the optimization models. If node m is the source of an anomaly, then $o^*_{nm}(t)$ is one for one m in $\{1,\dots,M\}$. This also concludes that $\gamma_{nm}(t)$ for the optimal m is larger than $\prod_{n'\in R_n}\gamma_{n'0}(t)$ because otherwise $o^*_{nm}(t)$ would not be one. This completes the proof.

Remark 9. If the condition in Remark 8 is satisfied for all n, then the network is not under the effect of any anomaly. This is an extremely important remark because it can help find the status of the entire network without running the optimization problems.

This remark is an immediate outcome of Remark 8, and thus the proof is not discussed.

Remark 10. For nodes n_1 and n_2 with the same set of sensors in their corresponding subnetworks R_{n_1} and R_{n_2}, the optimal solutions of $o_{n_1m}(t)$ and $o_{n_2m}(t)$ in Model 1 and Model 2 are the same.

This is a very important remark because it can be used to merge multiple nodes into one node and consequently lower the number of decision variables and constraints in both Model 1 and Model 2. This remark is particularly important for sparse networks where many nodes have no sensors, and thus they share the same sensor structures with at least one other neighbor node.

Proof. It is known that the optimal solution of any two decision variables is the same if the set of objective function and constraint parameters are the same. For Model 1 and 2, if sensors are the same for two nodes, then the corresponding inputs (i.e., network structure and sensor attributes) for the aggregation functions, the autoencoder, and the node-to-node anomaly detection algorithm are the same. As a result, the corresponding inputs of Model 1 and Model 2 are the same as well for these two nodes. This completes the proof.

5.5. An Optional Prescreening (PS) Phase (MLP$_0$)

Most real-life networks have a complex structure with a very large number of nodes. Also, network systems are often operating under normal conditions, with anomalies being relatively rare. Thus, analyzing all network nodes at every time epoch is not only computationally expensive but also unnecessary. To avoid searching the entire network for potential anomalies, we can use the topology of the network and sensor attributes in the entire network as inputs to train an MLP. Once the inputs are aggregated and transformed into the embedding space, we can feed them into the MLP with a binary output denoting whether there is an anomaly in the network. The outputs are organized so that any data point with anomalies, regardless of the location of the anomalies, is considered a positive class, and any data point where no single anomaly exists in the network is considered a negative class. In such an MLP, the data obtained from the entire network are considered one sample with a binary label. This can be simply done by defining an artificial node 0 where $R_o = \{1,\dots,N\}$. Once the model is trained, we can use it for future cases to determine whether there is an anomaly in the network. If the MLP predicts that the network has at least one anomalous node, then the detailed process of nodal and network anomaly detection is triggered. The role of this step in the framework is shown in Figure 7. We will also show with numerical experiments that adding such a preliminary step can significantly impact the computational time as well as the detection power of the algorithm.

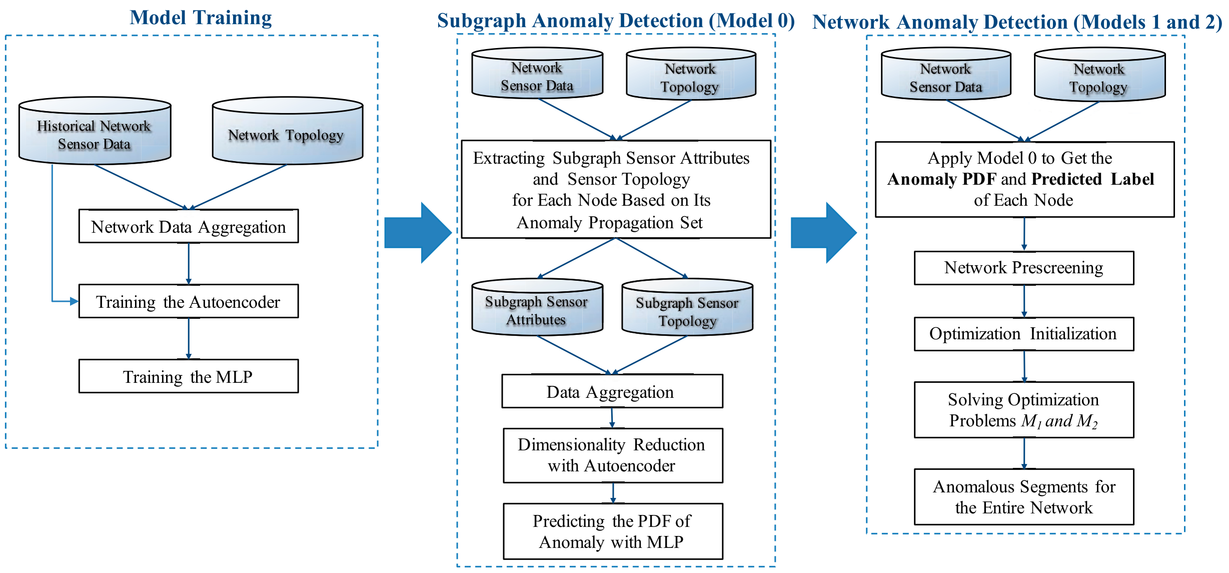
5.6. Detecting Anomalies in Connected Subgraphs

To extend the application of this work, we show how the proposed optimization models can be modified so that the set or sets of anomalous subgraphs with connected nodes are detected without considering propagation constraints. The formal definition of the problem is given below:

Problem B (Anomaly Detection in Connected Subgraphs). Given the topology of an attributed network, sensor locations, and sensor attributes, aggregate/analyze real-time network sensor observations collected from sensors across the network to locate (a) the most likely sources and types of potential anomalies (if any) and (b) the impacted anomalous nodes/subgraphs, subject to the connectivity constraints; that is, the detected anomalous subgraphs should be connected subgraphs.

In such a problem, the detected nodes do not need to follow predefined propagation paths. The only requirement is that the detected nodes to be connected. Because of this change, the aggregation of sensor data are not needed anymore because no predefined subgraphs exist in the model. The optimization model below can find one or multiple subgraphs under one or multiple types of anomalies, where the nodes within each anomalous subgraph are connected.

Figure 7. (Color online) An Overview of the Proposed Approach for Network Anomaly Detection ($\mathcal{M}_0$-$\mathcal{M}_2$)


Model Training
Historical Network Sensor Data
Network Topology
Network Data Aggregation
Training the Autoencoder
Training the MLP
Subgraph Anomaly Detection (Model 0)
Network Sensor Data
Network Topology
Extracting Subgraph Sensor Attributes and Sensor Topology for Each Node Based on Its Anomaly Propagation Set
Subgraph Sensor Attributes
Subgraph Sensor Topology
Data Aggregation
Dimensionality Reduction with Autoencoder
Predicting the PDF of Anomaly with MLP
Network Anomaly Detection (Models 1 and 2)
Network Sensor Data
Network Topology
Apply Model 0 to Get the Anomaly PDF and Predicted Label of Each Node
Network Prescreening
Optimization Initialization
Solving Optimization Problems M1 and M2
Anomalous Segments for the Entire Network

$\mathcal{M}_1^c$ **: Optimal Anomaly Detection with Connectivity Constraints:**

$$\max z = \sum_{n=1}^{N}\sum_{m=1}^{M} x_{nm}(t)\log\gamma_{nm}(t) + \sum_{n=1}^{N}\left[1-\sum_{m=1}^{M}x_{nm}(t)\right]\times\log\gamma_{n0}(t), \tag{17}$$

$$\text{Subject To :}\quad \sum_{m=1}^{M} o_{nm}(t)\le 1,\qquad \forall n\in\{1,\ldots,N\}, \tag{18}$$

$$o_{nm}(t)\le x_{nm}(t),\qquad \forall n\in\{1,\ldots,N\},\ \forall m\in\{1,\ldots,M\}, \tag{19}$$

$$x_{nm}(t)\le \sum_{n'} x_{n'm}(t)a_{n'n} + o_{nm}(t),\quad \forall n\in\{1,\ldots,N\},\forall m\in\{1,\ldots,M\}, \tag{20}$$

$$\sum_{n,m} o_{nm}(t)\le R,\qquad \forall n\in\{1,\ldots,N\},\ \forall m\in\{1,\ldots,M\}, \tag{21}$$

$$o_{nm}(t), x_{nm}(t)\in\{0,1\},\quad \forall n\in\{1,\ldots,N\},\ \forall m\in\{1,\ldots,M\}. \tag{22}$$

Constraint (18) ensures that each node can be the source of no more than one type of anomaly. Constraint (19) guaranties the causal relationship between o and x for each node. Constraint (20) ensures the connectivity between nodes, that is, each node is either the root node or it is connected to one of its immediate parents. Together, Constraints (19)–(20) can control the propagation of anomalies so that an anomaly at any node may be propagated to one of the node's immediate neighbors, which are defined through the elements of the adjacency matrix **A**. Finally, Constraint (21) limits the number of root nodes in the detected subgraphs to a maximum threshold R defined by the user. The value of R dictates the maximum number of anomalies that can be detected in the network. This model is more complex than the optimization models $\mathcal{M}_1$ and $\mathcal{M}_2$ because it has more variables and constraints. We suggest using this model only if (a) there are no available anomaly propagation paths and (b) connectivity between anomalous nodes is required. For a tree structure, because each node has only one parent, by setting $R = 1$ we can force the model to choose only a connected subgraph with one root node and potentially any size. For nontree structures, the above model does not guarantee that all subgraphs are connected to each other; however, it ensures that the nodes in each detected anomalous subgraph are connected. This is a limitation over models that detect only one arbitrarily shaped subgraph with connected nodes. The objective function is to maximize the joint probability of all nodes' health status. The solution of this optimization problem is the same as the solution of the GraphScan method (when applied on a DAG) with an optimal neighborhood size. In Section 6, we provide a comprehensive example to show how the outcomes of Problem A (and its extensions) and Problem B can result in two different solutions.

5.7. Overview of the Real-Time Implementation of the Proposed Framework

To better understand the relationship between the elements of the proposed framework, the main steps of

Table 2. Important Statistics for the Network Topology of the Two Data Sets

Data set	Nodes	Edges	Density	Maximum degree	Minimum degree	Average degree
Data set 1	1.1K	1.5K	0.002254	17	1	2
Data set 2	10K	10K	0.0001	440	1	2

the anomaly detection process at a sample time instance are shown in Figure 7. Algorithm 2 in Appendix D in the online supplement also summarizes the technical details of the implementation phase.

6. Numerical Experiments

Because of the structure of power distribution networks and the extensive availability of binary sensor data, they are perfect examples of where the proposed frameworks can be applied. A comprehensive set of numerical experiments inspired by real power distribution networks in the state of Florida is designed to show the effectiveness and application of our framework. We also show the application of our work to detecting contamination in a water distribution network and compare our framework with a strong model in the literature. We will demonstrate how our approach can detect anomalies impacted by a set or sets of connected nodes without having to follow predetermined propagation paths. Finally, we provide experiments on the scalability of our anomaly detection approach with various sizes of simulated networks. All numerical experiments are conducted using R on a iMac with a 3.5 GHz Intel i9-9920X processor and 32 GB of RAM. A summary of the data used in this article can be found in Appendix E in the online supplement.

6.1. Detecting Anomalies in Power Distribution Network Structures

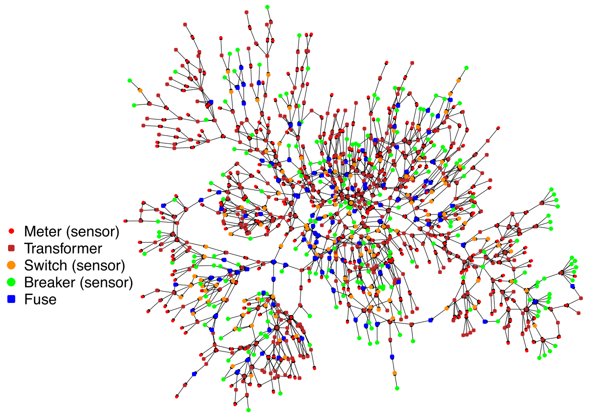
Two publicly available topology data sets related to power distribution networks are used for the first part of our numerical experiments. The network topologies in both data sets have a grid structure that starts at the top from the substation and moves down to the lowest level (customers/meters). Some of the topological characteristics of these networks are given in Table 2. In the smaller data set (data set 1), the network has 1,138 nodes out of which there are 851 sensor nodes, whereas in the larger data set (data set 2), the network has 10,090 nodes out of which there are 7,001 sensor nodes. For both networks, the power is transformed from the substation into the meters through different nodes. Each node may generate some sensor attributes. Because sensors are placed only at a randomly selected subset of nodes, we have a heterogeneous attributed network. A simple view of the network in data set 1 is shown in Figure 8.

6.1.1. Network Data Set 1. This data set is for a 1,138-bus power network provided in a Network Repository (Rossi and Ahmed 2015). Some of the characteristics of this network are given in Table 2. We divided all nodes into one of five types: feeders/breakers, switches, fuses, transformers, and meters.

6.1.2. Network Data Set 2. This network data set, created using the U.S. Reference Network Model (NREL 2016), is a fully synthetic distribution data set for the San Francisco Bay Area and has been created using the U.S. Reference Network Model. We selected a part of the network topology with 10,090 nodes and then randomly selected some nodes to be the hosts of sensors.

6.1.3. Anomalies Across the Networks. The propagation of anomalies in distribution networks depends entirely on the structure and topology of the distribution network. For example, if a nonmomentary fault occurs in an overhead transformer, the fuse at the transformer can isolate the anomaly so that only downstream customers are affected by the anomaly. Similarly, if a transformer has an issue, then all premises downstream of the transformer will experience an outage. We use the same rules for all nodes to define the set of anomaly propagation paths. In Figure 9, we show six examples from data set 1 of how an anomaly in one node is propagated to its neighborhood

Figure 8. (Color online) A Simple Overview of the Power Network in Data Set 1



Notes. Each node may be a host for a sensor or a physical device. Circles are used to represent sensor nodes.

according to its anomaly propagation set. We can segment the network into seven clusters (six anomalous clusters and one healthy cluster). For both data sets, two types of anomalies are considered. A total of 1,000 time instances are considered in which anomalies are detected at each time instance independently of others. For each network and at each time instance, we assumed a 50% chance of being under at least one anomalous event. If the network is anomalous, then a random number between 1 and 50 is generated to determine the number of total anomalies in the network. Also, the types and the locations of anomalies for anomalous points are randomly found. The final outcome of the simulation for each network and each timestamp is the set of anomalous nodes with their types, the original locations of anomalies, and network sensor data.

6.1.4. Network Sensor Data. A total of 40 binary sensor attributes are considered (i.e., $J = 40$). Sensor attributes generated at different nodes depend on the type of the node, and no node can generate all types of sensor attributes. We have made a realistic assumption that sensor types can have their own unique set of sensor attributes and also have attributes that are generated by other sensor types. For instance, breakers and switches in data set 1 generate sensor attributes 23–33. Examples of such binary signals are various types of Supervisory Control and Data Acquisition errors generated across the network over time. Although this article does not require a specific stochastic distribution for sensor data, to conduct numerical experiments, it is assumed that the jth sensor attribute at node n follows a conditional Bernoulli distribution given the status of node n ($\boldsymbol{x}_n(t)$) where the sensor is located, that is,

$$P(y_{nj}(t) = 1 \,|\, \boldsymbol{x}_n(t)) = p_{0,j}^{\left(1-\sum_{m=1}^{2} \boldsymbol{x}_{nm}(t)\right)} \prod_{m=1}^{2} p_{m,j}^{x_{nm}(t)}, \quad (23)$$

where parameters $p_{0,j}$ and $p_{m,j}$ are randomly generated from the following uniform distributions:

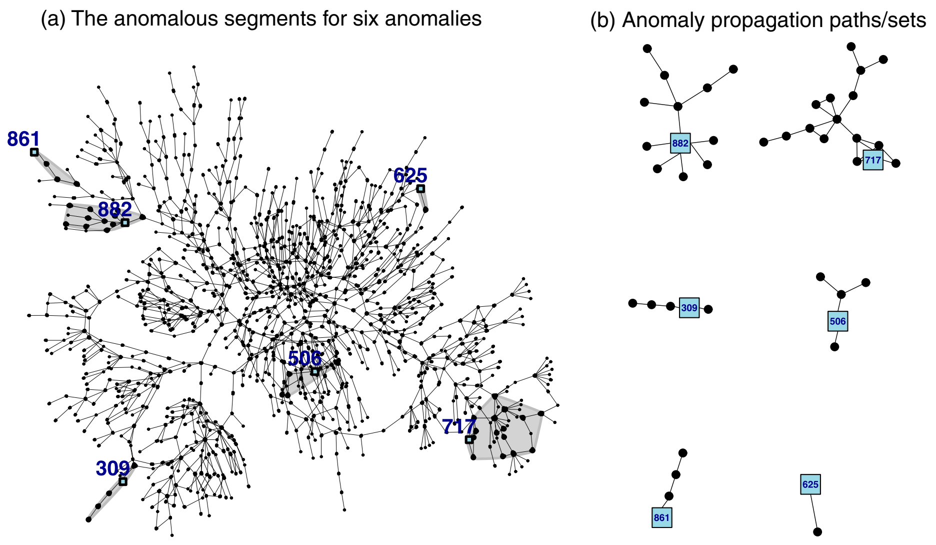
$$p_{0,j} \sim \mathrm{U}(0, 0.15), \quad p_{m,j} \sim \mathrm{U}(0, 0.45),$$
$$\forall j \in \{1, \ldots, 40\}, m \in \{0, 2\}.$$

Note that $p_{0,j}$ and $p_{m,j}$ can be defined as the false positive rate (FPR) and true positive rate (TPR) for the mth type of anomaly, respectively.

6.1.5. Summary of Training, Testing, and Validation Sets. For each data set, we simulated 1,000 time instances/networks out of which 509 networks in data set 1 and 703 in data set 2 have at least one anomaly event. We considered 500 networks for training and the other 500 for testing. We considered 100 out of the 500 training samples as a validation data set to tune in hyperparameters.

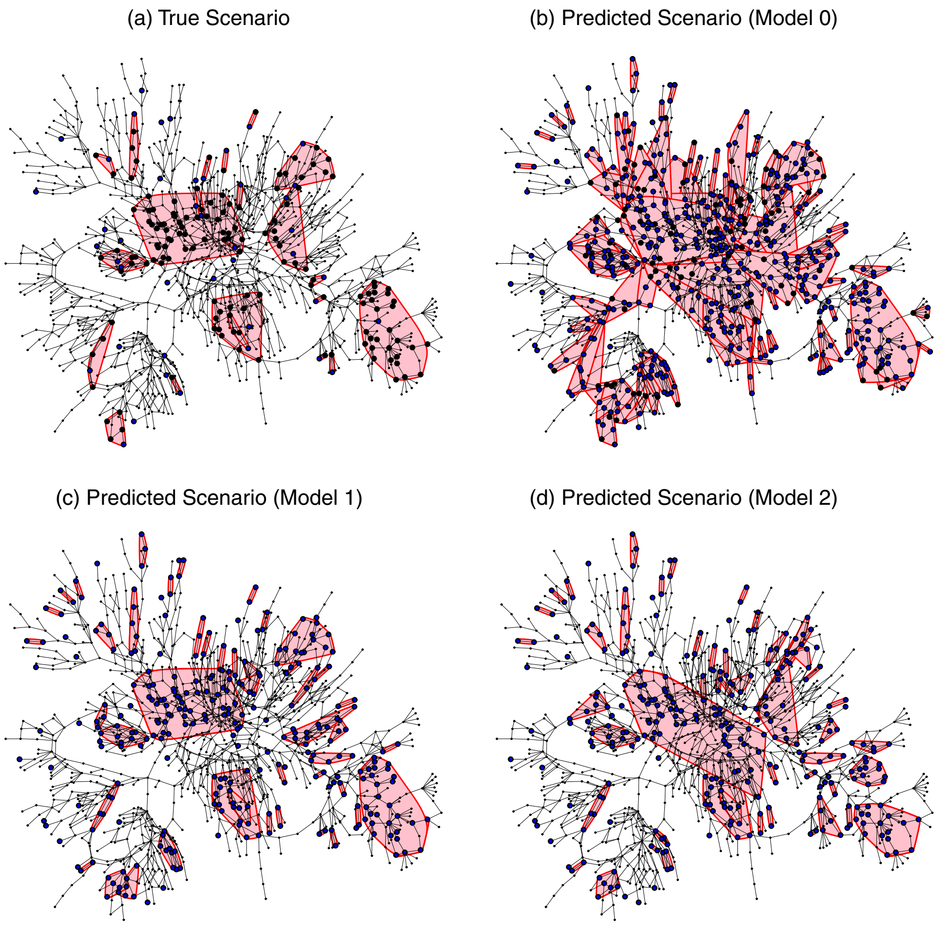
6.1.6. An Example of Network Anomaly Detection. As discussed earlier, the outcomes of MLP may be directly used for node-level anomaly detection ($\mathcal{M}_0$). Also, such outcomes can be used as inputs for $\mathcal{M}_1$ and $\mathcal{M}_2$ to find the location of anomalies in an integrated manner. To better present the difference between these approaches, we provide an example of an anomaly scenario in Figure 10 and show how $\mathcal{M}_0$, $\mathcal{M}_1$, and $\mathcal{M}_2$ can predict anomalies and all impacted anomalous nodes. The true scenario in this example has 43 anomalies

Figure 9. (Color online) A View of a Network with Six Simultaneous Anomalies and Segmented Subnetworks


(a) The anomalous segments for six anomalies
861
882
625
506
717
309
(b) Anomaly propagation paths/sets
882
717
309
506
861
625

Note. The square nodes are the original locations/sources of anomalies and the filled circles are the impacted nodes based on the propagation paths of the source nodes.

Figure 10. (Color online) A View of True vs. Predicted Anomalous Nodes


(a) True Scenario
(b) Predicted Scenario (Model 0)
(c) Predicted Scenario (Model 1)
(d) Predicted Scenario (Model 2)

Notes. Impacted and predicted nodes have larger vertex size than healthy nodes. Anomalies with more than 1 node are segmented in pink for better visibility.

with a total of 184 impacted nodes. It is clear from Figure 10 that Model 0 ($\mathcal{M}_0$), which is a node-level anomaly detection approach, is subject to a large number of healthy nodes that are predicted as anomalous. However, Model 1 and Model 2 perform better in predicting impacted nodes with fewer false alarms. Models 1 and 2 also can cluster anomalies better than Model 0. This is mainly because these models predict anomalies for the entire network in an integrated and dependent manner. Most of the false alarms for Models 1 and 2 are for small segments and single node anomalies.

6.1.7. Summary of the Results for Network Anomaly Detection ($\mathcal{M}_1$ and $\mathcal{M}_2$).

To evaluate the performance of the network anomaly detection models ($\mathcal{M}_1$ and $\mathcal{M}_2$) discussed in Section 5 and the subgraph anomaly detection $\mathcal{M}_0$ discussed in Section 4.4, we applied them to the testing sets of both data sets. Then, for the three cases of no anomaly, anomaly type 1, and anomaly type 2, we checked to see how each model predicted the true labels of all nodes. To better evaluate the outcomes of the proposed models, we compare them with four relevant baseline models/competitors.

Table 3. Optimal Hyperparameters for MLP in the Proposed and Baseline Models

Model name	Layer 1 nodes	Layer 2 nodes	Dropout rate 1	Dropout rate 1	Optimizer	LR annealing rate
$\mathcal{B}_1$	8	2	0.3	0.3	Adam	0.05
$\mathcal{B}_2$	16	8	0.3	0.3	Adam	0.1
$\mathcal{B}_3$	8	4	0.2	0.3	Adam	0.1
$\mathcal{M}_0$	16	8	0.2	0.3	Adam	0.05
$\mathcal{M}_1$	8	4	0.2	0.3	Adam	0.1
$\mathcal{M}_2$	8	8	0.2	0.3	Adam	0.05

The first natural competitor would be a model that uses only raw sensor data at each node directly as inputs and builds a three-label classifier that can predict the label of each node. This baseline competitor, referred to as MLP with raw sensor data or $\mathcal{B}_1$, is the most straightforward way of using network sensor data for anomaly detection. The second baseline competitor, referred to as MLP with aggregated sensor data or $\mathcal{B}_2$, is a model that employs the aggregate sensor data as the input for a multiclass classifier where only sensor data are aggregated for each subnetwork and then used as the classifier inputs. Applying this model helps evaluate the effect of including sensor topology for anomaly detection. The third competitor, referred to as MLP with aggregated sensor and topology or $\mathcal{B}_3$, is a model that aggregates both sensor data and network topology and uses them as inputs for a three-label classifier. Such data are actually the inputs of the autoencoder discussed in Section 4.2. Note that in model $\mathcal{M}_0$, we transform the fully aggregate sensor and network data into the new embedding space using the autoencoder and then use the embedded data as inputs for the classifier for predicting node labels using the trained MLP. This model can also be considered as a baseline for our final proposed models $\mathcal{M}_1$ and $\mathcal{M}_2$. Table 3 summarizes the optimal structures of these baseline models with regard to the hyperparameters (found with a grid search).

We selected six commonly used and reasonable performance metrics to evaluate the performance of each model. These metrics are the false positive rate (false alarm rate), which indicates the percentage of healthy nodes incorrectly detected as anomalous; true positive rate (detection rate, sensitivity, recall), which indicates the percentage of anomalous nodes that have been properly detected; precision (positive predictive value), which indicates the percentage 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 (one means perfect agreement and zero 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 metrics, we compute the type detection rate, which measures the correct identification percentage for each type of anomaly. Results shown in Table 4 can be summarized as follows. First, it can be seen that using only raw network sensor data as inputs results in a relatively poor performance for detecting anomalies. This is mainly because of the ignorance of the heterogeneity of the network data and topology and the fact that nodes and their corresponding subnetworks are analyzed independently. However, as we improve the data fusion process and include the dependencies between nodes, the performance of the models improves. For instance, almost all performance metrics of models $\mathcal{B}_1$ and $\mathcal{B}_2$ are worse than model $\mathcal{B}_3$. Second, $\mathcal{M}_1$ and $\mathcal{M}_2$ provide better results compared with all other baselines as shown by larger values in some of the overall performance measures (i.e., F-score, overlap coefficient, and type detection rate). Also, greater precision in the proposed models $\mathcal{M}_1$ and $\mathcal{M}_2$ indicates their better performance in minimizing false alarms, which is a key metric for decision makers in the power utility industry. Third, better performance measures in $\mathcal{M}_0$ compared with $\mathcal{B}_3$ indicate that the autoencoder can slightly improve the performance of the task of anomaly detection possibly because of its power to lower the complexity of the problem. The results also show no significant difference between $\mathcal{M}_1$ and $\mathcal{M}_2$.

In summary, we can conclude that the two integrated optimization-based anomaly detection methods that take the status of the entire network into account perform better at the task of anomaly detection compared with their natural competitors. Similarly, with regard to predicting the correct type of anomaly, optimization-based models perform better. We justify this performance with the fact that they

Table 4. Performance of the Proposed and Baseline Modes in the Two Data Sets

	Network data set 1						Network data set 2					
Model type	FPR	TPR	Precision	F-Score	Overlap coefficient	Type detection	FPR	TPR	Precision	F-Score	Overlap coefficient	Type detection
Model $\mathcal{B}_1$	1.6	56.1	72.8	63.4	46.4	51.0	1.1	99.2	49.6	66.1	49.4	85.6
Model $\mathcal{B}_2$	1.7	56.4	71.9	63.2	46.2	85.6	1.5	99.5	43.0	60.1	42.9	85.7
Model $\mathcal{B}_3$	1.3	56.7	76.7	65.2	48.4	85.7	1.1	99.4	49.5	66.1	49.4	85.5
Proposed Model $\mathcal{M}_0$	6.5	98.1	53.4	69.2	52.9	88.2	0.9	99.1	54.7	70.5	54.4	82.2
Proposed Model $\mathcal{M}_1$	2.2	91.6	75.9	83.1	71.1	92.9	0.8	99.4	57.7	72.9	57.5	90.4
Proposed Model $\mathcal{M}_2$	2.4	92.6	74.2	82.4	70.1	91.7	1	99.9	51.9	68.3	51.8	96.1
$\mathcal{M}_0$, no autoencoder	7.4	98	50.3	66.5	49.9	88.7	1.1	99.4	49.5	66.1	49.4	85.5
$\mathcal{M}_1$, no autoencoder	2.4	90.3	74.4	81.6	68.9	92.7	1.1	99.5	51.4	67.8	51.3	93.1
$\mathcal{M}_2$, no autoencoder	2.6	91.5	72.7	81.1	68.2	92.3	1.1	99.8	50.4	66.9	50.3	96.9

both analyze the entire network at once; as a result, the reduction of all nodes is better in line with the topology of the grid and the corresponding anomaly propagation sets. It should be pointed out that the overall efficiency and the relatively better performance of the proposed models compared with their natural competitors depend on many factors, including the complexity of the network topology and its size, the power of sensor attributes represented by high TPR and Low FPR, and potentially the number of anomalous subsets in the network.

6.2. The Impact of Optional Dimensionality Reduction on Sensor Information Loss

To measure the impact of the optional unsupervised dimensionality reduction step (autoencoder) on $\mathcal{M}_1$ and $\mathcal{M}_2$, we considered the original aggregated data without using the autoencoder as inputs for the optimization models and then calculated the same performance measures as Section 6.1. By changing the code size (the number of nodes in the bottleneck layer), the number of layers, and the number of nodes per layer, we conduct grid search experiments to determine the set of hyperparameters. All other parameters of the autoencoder are set as default in the R autoencoder-package (loss.type: squared, activ.functions: tanh, optim.type: adam, n.epochs = 80). The final encoder has only one hidden layer (embedding space) with eight nodes (neurons). In other words, the encoder takes the inputs and maps to an eight-dimensional layer. Comparing the results in rows 4–6 and rows 7–9 of Table 4 implies that the information loss of the dimensionality reduction phase was not significant; thus, the performance of the models after reducing the feature size stays almost unchanged. In fact, some metrics slightly improve, which may be because the complexity of the models decreases after using the dimensionality reduction step. Thus, MLP models are better trained. Because the autoencoder transforms the 80-dimensional sensor-topology vector to an 8-dimensional vector in the embedded space, it can help reduce the training time for the MLPs to find the probability status of each node. On average, using the autoencoder could save 0.24 seconds per network sample in the training phase of a single MLP in data set 1 (14.1% improvement) and 3.2 seconds in data set 2 (15.3% improvement); this takes into account the time necessary to train the autoencoder. This impact may be more significant for cases with more training data, networks of larger size, and networks with more sensor attributes. This step has no direct impact on the CPU time in the testing set, particularly for the optimization models in $\mathcal{M}_1$ and $\mathcal{M}_2$ where both models use a one-dimensional outcome obtained from trained MLPs as inputs.

6.3. The Effect of the Preoptimization Steps on Computational Complexity

In this subsection, we present the effect of Remarks 7 and 8 with regard to their ability to lower the complexity of the optimization models. The impact of Remark 7 on $\mathcal{M}_1$ and $\mathcal{M}_2$ for each network sample is a reduction in the number of decision variables by $\frac{100}{M}$%, that is, from 2,276 ($M * N$) to 1,138 ($M * N - N$) in data set 1 and from 20,180 to 10,090 in data set 2. However, the effect of Remark 8 is different on $\mathcal{M}_1$ and $\mathcal{M}_2$ and also cannot be found analytically. For $\mathcal{M}_1$ and $\mathcal{M}_2$, the already reduced numbers of variables after Remark 7 averaged over all 500 training sets are changed to 327 and 343, respectively, for data set 1 (around 70% reduction) and 2,623 and 2,513 for data set 2 (around 74% reduction). This reduction also depends on how many nodes are anomalous in the network, that is, the greater the number of healthy nodes/subgraphs are in the network, the greater the reduction in the number of decision variables. In summary, the average reduction in the number of variables in $\mathcal{M}_1$ and $\mathcal{M}_2$ after implementing Remarks 7 and 8 was about 85.6% and 84.9% for data set 1 and 87% and 87.6% for data set 2. This is a significant computational benefit, particularly for large networks with a complex topology and sensor structures. The average CPU time needed to monitor the entire network in our numerical experiment, including all steps in the testing phase, was 5.5 seconds for data set 1 and 17.2 seconds for data set 2. Without Remarks 7 and 8, that number was around 67 seconds for data set 1 and 274 seconds for data set 2. The effects of Remarks 9 and 10 were both less than 2%. Such effects mainly depend on the network topology and how anomalies are simulated.

6.4. The Impact of the Prescreening Phase on Network Anomaly Detection

As discussed earlier, the idea behind the prescreening phase originated from the fact that real networks are often at a normal state and anomalies are relatively scarce. Thus, there is often no need to continuously check the entire network for the task of anomaly detection. To control the task of anomaly detection, we can train a single MLP where its inputs are the data from the embedding space. In other words, each sample is an integration of all sensor data and the entire network sensor topology. The label of each sample is binary reflecting whether the network is under an anomaly condition with at least one anomalous node. Similar to all MLPs discussed in the paper, the hyperparameters can be tuned in with the grid search. Once the corresponding MLP is trained, it can be first used to determine whether a network has an anomaly. If the prediction is positive, then we can apply models $\mathcal{M}_1$ and $\mathcal{M}_2$ to find the location of anomalies. Otherwise, we can conclude that the network has no anomalous node. The potential drawback of this prescreening phase is when the model mistakenly labels an anomalous network as one without any anomaly. We applied the trained prescreening MLP on the testing sets for both data sets and computed important performance measures. In Table 5, we report the improvement in important performance metrics after using this step for both $\mathcal{M}_1$ and $\mathcal{M}_2$ and in both data sets. It can be seen that the PS step can significantly lower the number of nodes falsely reported as anomalous. Also, as expected, applying the prescreening step resulted in a slightly lower detection rate of anomalous nodes. This negative performance change is significantly lower than the improvement in false alarms. Results shown in Table 5 verify the better performance of the PS in reducing false alarms (see for instance the change in the FPR) and increasing precision, which together help slightly improve the overlap coefficient. Because this step ignores the type of anomaly, it has no impact on the ability of Models 1 and 2 to detect the type of anomaly.

6.5. Comparison with GraphScan to Detect Contamination in Water Networks

In this subsection, we compare our work with one of the well-known scan statistics models developed for network anomaly detection, namely, the GraphScan model by Speakman et al. (2015). The objective of the GraphScan model is to find the set of connected subgraphs with the highest anomaly scores. With a set of numerical experiments, we demonstrate when the GraphScan model performs better/worse than our proposed model $\mathcal{M}_1$ and its extension $\mathcal{M}_1^c$. In all of our numerical experiments, each network is subject to only one anomaly. This is because the GraphScan model is not designed to find multiple anomalous subgraphs or the case where there is no anomaly at all in the network. It is known that the dispersion and propagation of the contamination in a water distribution network depend on the topology of the network and the point (sources) of contamination. In other words, contamination at a point in the network can only affect downstream locations in the network. This makes detecting contaminations in water distribution networks a perfect case for applying our model and the GraphScan model. For comparison, we used two data sets related to water distribution networks. The first topology is obtained from Georgescu (2012), which is a small network with 43 nodes and 78 edges. The second network, which has 130 nodes and 180 edges, is related to the well-known Battle of the Water Sensor Networks data set, which is a benchmark data set originally used for sensor location optimization. For both networks, the adjacency matrix is used to build the graph topology and define the anomaly

Table 5. Classification Measures for $\mathcal{M}_1$ and $\mathcal{M}_2$ Before and After the Prescreening Step

	Network data set 1						Network data set 2					
Model type	FPR	TPR	Precision	F-score	Overlap coefficient	Type detection	FPR	TPR	Precision	F-score	Overlap coefficient	Type detection
$\mathcal{M}_1$ with PS	2.0%	91.3%	78.1%	84.2%	72.7%	92.9%	0.8%	99.1%	58.4%	73.5%	58.1%	90.4%
$\mathcal{M}_2$ with PS	2.2%	92.3%	76.2%	83.5%	71.6%	91.7%	1.0%	99.6%	52.6%	68.9%	52.5%	96.1%
Improvement for $\mathcal{M}_1$	−11.8%	−0.3%	2.8%	1.3%	2.3%	0.0%	−3.7%	−0.3%	1.3%	0.7%	1.1%	0.0%
Improvement for $\mathcal{M}_2$	−10.2%	−0.3%	2.7%	1.3%	2.3%	0.0%	−2.9%	−0.3%	1.4%	0.9%	1.3%	0.0%

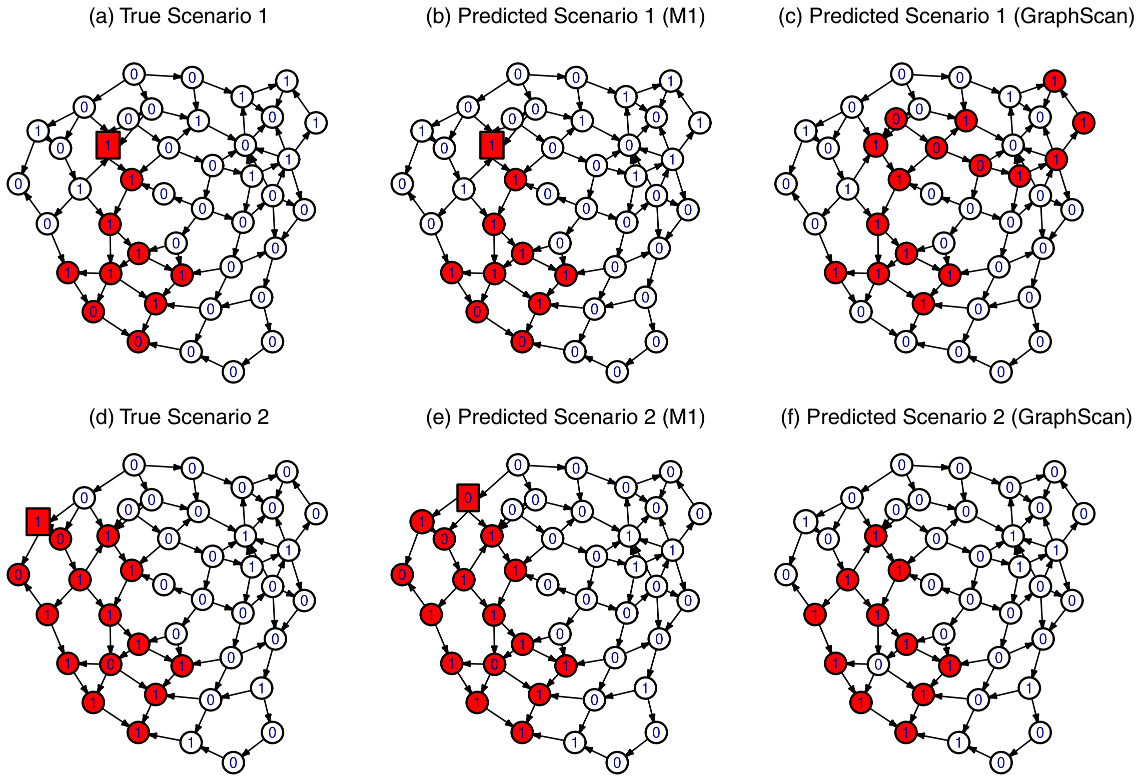
propagation paths. For all numerical experiments, we assume that the flow patterns do not change and thus the topology of the networks is fixed. Similar to Speakman et al. (2015), we assume that binary sensors are located at each of the pipe junctions/graph nodes. For sensor data generation, we assume a fixed false positive rate and true positive rate for independent sensors so that the expectation-based binomial (EBB) scan statistic can be used as the scoring function for the GraphScan approach to find the most positive connected subgraph. The score of each node based on the EBB GraphScan method is as follows:

$$\text{Score}_i = x_{n1}(t)\log\left(\frac{\text{TPR}}{\text{FPR}}\right) + (1 - x_{n1}(t)) \times \log\left(\frac{1-\text{TPR}}{1-\text{FPR}}\right). \tag{24}$$

We should point out that the comparison was carried out with an optimization-based GraphScan method, where the neighborhood size is assumed to be optimal. This means that the results presented here may be different from the heuristic model developed in Speakman et al. (2015). It should also be noted that the size of the neighborhood (or radius of the neighborhood) has a significant impact on the scan statistic model's detection power and spatial accuracy. In systems where anomalies can have a wide range of neighborhood sizes, there is frequently no prior knowledge on the expected size of the events of interest. As a result, the use of such scan statistics methods is restricted.

With regard to water contamination (anomaly), we consider that when a contaminant is inserted from one node, then it propagates to other nodes according to two possible contamination scenarios of (a) full propagation, that is, all downstream nodes are contaminated, or (b) partial propagation, that is, only a subset of nodes connected (directly or indirectly) to the source of contamination is contaminated. In Figure 11, we show how our proposed model $\mathcal{M}_1$ and the GraphScan method can predict two cases of anomalies, referred to as Scenario 1 and Scenario 2, on water network 1, when the anomalies are fully propagated in the network. Also, the output of sensors located at each node is shown inside each vertex. For the true Scenario 1, an anomaly occurs at the node shown by a rectangle and influenced all nine downstream nodes. It can be seen from the first row of this figure that our model ($\mathcal{M}_1$) can detect all anomalous nodes. However, the GraphScan model ended up selecting more nodes as anomalous simply because there is a cluster of false positive sensors close to the original cluster of anomalies that were incorrectly selected as anomalous. Also,

Figure 11. (Color online) A View of True vs. Predicted Anomalous Nodes for the Case of Full Propagation and for Two Scenarios of Anomalous Networks


(a) True Scenario 1
(b) Predicted Scenario 1 (M1)
(c) Predicted Scenario 1 (GraphScan)
(d) True Scenario 2
(e) Predicted Scenario 2 (M1)
(f) Predicted Scenario 2 (GraphScan)

two of the true anomalous nodes are incorrectly classified as normal because of their inactive (false negative) sensors. In Figure 12, two other anomalies are considered for the case of partial propagation. For both examples, the GraphScan model outperformed our model $\mathcal{M}_1$. We have repeated the experiments on 1,000 networks considering three conditions for sensor FPR and TPR and then reported the three performance measures of FPR, TPR, and overlap coefficients in Table 6.

In summary, we can conclude that in the case of the full propagation, although GraphScan is able to correctly include contaminated sensors that did not trigger (false negatives) to connect clusters of true positives, it can exclude the same sensors if they are in the boundaries of the connected graphs. Also, GraphScan can incorrectly include all false positive sensors if they are connected to the anomalous subgraphs. Unlike our proposed model $\mathcal{M}_1$, GraphScan cannot consider propagation constraints embedded in many networks, such as power distribution networks. Also, it is not designed to find the source of the anomaly. In the case of the partial propagation, GraphScan works significantly better than our model $\mathcal{M}_1$ because it is not forced to follow propagation rules and include true negatives in the anomalous nodes. Another interesting observation is that as the power of sensors with regard to anomaly detection degrades (i.e., TPR decreases and FPR increases), the superiority of model $\mathcal{M}_1$ increases for the case of full propagation. For the case of partial propagation, the performance of our model approaches that of GraphScan. As expected, results also verify that the model extension $\mathcal{M}_1^c$ provides the same results as the GraphScan method. This is because the score of each node in the objective function of $\mathcal{M}_1^c$ converges to the score function in Equation (24), and then both models become equal. For the case of partial propagation and very large networks, although the model extension $\mathcal{M}_1^c$ works the same as GraphScan, it is slower because of the existence of the optimization model.

6.6. Scalability Analysis

To evaluate the scalability of the model with respect to the size (N), the degree (average number of edges per node), and the topology of the network, we conducted a set of experiments on randomly generated graphs of six types of well-known and widely used directed acyclic graph topologies, where each has reportedly many real-word applications and are suitable for the study of complex networks (Durrett 2006). These six types of networks are regular DAG (a DAG graph where every node has about the same degree), power law DAG (a scale-free graph where the degree distribution is heavy tailed, that is, many low degree nodes and a few high degree nodes), Barabási-Albert DAG (a graph with power-law degree distribution with a preferential attachment mechanism, which means the more connected a node is, the more likely it

Figure 12. (Color online) A View of True vs. Predicted Anomalous Nodes for the Case of Partial Propagation and for Two Scenarios of Anomalous Networks

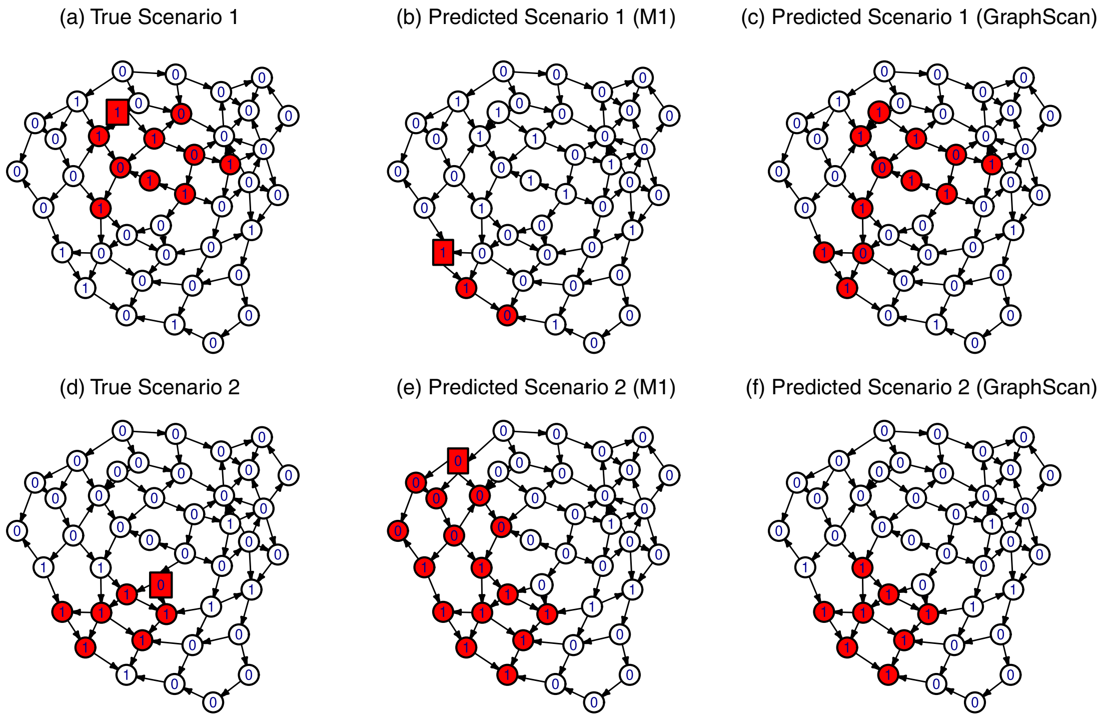
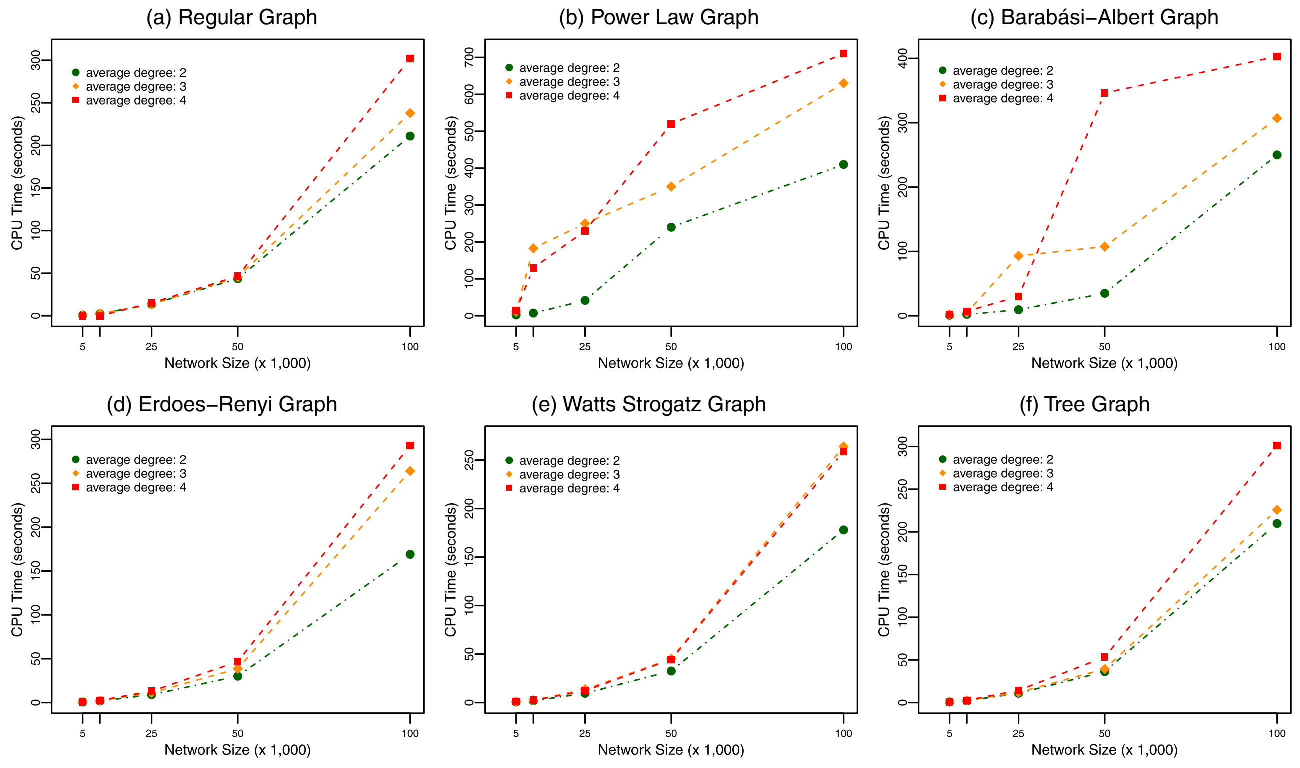

(a) True Scenario 1
(b) Predicted Scenario 1 (M1)
(c) Predicted Scenario 1 (GraphScan)
(d) True Scenario 2
(e) Predicted Scenario 2 (M1)
(f) Predicted Scenario 2 (GraphScan)

Table 6. The Comparison Between Our Proposed Model, Its Extension, and Modified GraphScan Under Various Scenarios for Sensor Output and Anomaly Propagation

		Sensor information -->	Sensor TPR = 0.9, FPR = 0.1			Sensor TPR = 0.8, FPR = 0.2			Sensor TPR = 0.7, FPR = 0.3		
Data set	Anomaly propagation scenario	Model type	FPR	TPR	Overlap coefficient	FPR	TPR	Overlap coefficient	FPR	TPR	Overlap coefficient
Water Network 1	Full propagation	Model $\mathcal{M}_1$	2.0	99.0	92.0	7.2	91.5	76.0	14.5	83.9	60.0
		Model $\mathcal{M}_1^c$	1.2	90.6	86.6	12.4	72.3	53.6	28.6	63.7	35.8
		EBP GraphScan	1.2	90.6	86.5	12.4	72.3	53.6	28.6	63.7	35.8
	Partial propagation	Model $\mathcal{M}_1$	16.2	68.8	48.2	21.0	69.1	44.8	27.2	66.9	39.7
		Model $\mathcal{M}_1^c$	3.8	89.1	80.9	13.3	79.9	59.4	27.6	71.1	41.9
		EBP GraphScan	3.9	89.2	80.9	13.3	79.9	59.4	27.6	71.1	41.9
Water Network 2	Full propagation	Model $\mathcal{M}_1$	1.3	99.0	94.4	3.6	96.4	84.6	5.1	93.2	77.6
		Model $\mathcal{M}_1^c$	1.2	90.3	86.4	5.1	80.9	67.5	13.0	70.2	46.2
		EBP GraphScan	1.2	90.4	86.4	5.1	80.9	67.5	13.0	70.2	46.2
	Partial propagation	Model $\mathcal{M}_1$	4.5	50.8	38.6	6.4	56.8	39.4	11.5	56.1	30.9
		Model $\mathcal{M}_1^c$	1.2	91.0	84.0	5.4	79.7	58.1	14.1	67.1	33.5
		EBP GraphScan	1.2	91.0	83.9	5.4	79.7	58.1	14.1	67.1	33.5

is to receive new links), Erdoes-Renyi DAG (a random graph that connects two nodes based on the edge probability or the edges selected uniformly at randomly), Watts Strogatz DAG (a graph that interpolates between the regular and Erdoes-Renyi graph), and tree/hierarchical DAG (a graph that follows a tree structure with no nodes having more than one parent). We used the RandDAG package in R to generate random DAG graphs. We also considered three different cases for the expected number of neighbors for each node to verify the scalability of the model with regard to the average degree of the network for each type of graph structure. To study the effect of network size, the experiments were conducted for five scenarios (around 5,000 nodes, 10,000 nodes, 25,000 nodes, 50,000 nodes, and 100,000 nodes). For each combination of graph topology, number of nodes, and degree, we simulated random anomalies in 100 graphs (50 for training and 50 for testing) and applied our model to detect anomalies in the testing set. For each experiment, the average CPU time (time it takes to find anomalies) per graph based on Model $\mathcal{M}_1$ was reported in Figure 13. We also considered three different cases for the number of anomalous subgraphs to see the effect of the number of anomalous nodes on the time it takes to detect anomalies. Also, we repeated the experiments for $\mathcal{M}_2$ and for all of the above conditions. Because no significant changes were observed for the last two sets of experiments, corresponding experiments are not reported here. All experiments were conducted on an iMac workstation with 3.4 GHz Intel Core i5 and 32 GB of RAM. As expected, the run time increases as the connectedness and complexity of the network increase, but the model is still scalable for $n \leq 100{,}000$ except for cases with higher average neighbors (i.e., three and four) in the power law and the Barabási-Albert graphs whose degree distributions follow a power law. In fact, we found that in the randomly generated graphs, there are nodes at the top layers of the graph with a very large degree. Such a large number of neighbors for nodes is not realistic for many systems, such as power and water distribution networks where nodes often have a small set of neighbors and nodes with higher degrees are located at the lower layers of the network. For instance, almost all eight power grid data sets reported in a publicly available network repository (Rossi and Ahmed 2015) have less than 5,000 nodes, a maximum degree less than 20, and an average degree around two. Similarly, out of the 22 real water distribution networks having different sizes reported in Giustolisi et al. (2017), all have a size less than 30,000 nodes, a maximum degree of less than 11, and an average degree around two. The reason that our model performed slower for large power law and Barabási-Albert graphs was mainly because of the

Figure 13. (Color online) Performance of the Anomaly Detection Model on Random Graphs of Varying Size, Average Degree, and Topology



high number of propagation paths and complexity caused by the power law degree distribution, which make the proposed preoptimization steps (discussed in Section 5.4) less effective. Although anomalies can be detected faster in simple structures (such as trees) and smaller networks, the proposed framework is still scalable on nontree networks.

Our framework could not be tested on networks larger than 100,000 nodes (125,000 nodes and above) mainly because of the memory requirement of forming the problem, that is, generating the adjacency and propagation matrices and setting up the corresponding optimization models. The current scalability of the model for networks smaller than size 100,000 (Figure 13) makes it highly applicable for anomaly detection in a wide range of contexts in real-world networks as reported in the literature, such as power distribution networks (where each substation is considered a graph), water distribution networks, and communication networks where reportedly the numbers of nodes are within the feasibility range of our framework. Also, compared with similar approaches, such as well-known GraphScan statistics algorithms reported in Speakman et al. (2015), the proposed framework is highly competitive as it finds anomalies in an optimal manner and has a reasonable running time for various graph complexities and sizes. To use our frameworks for complex and large networks (e.g., networks with more than 100,000 nodes and large average degrees), users should utilize high-performance computing, storage, and memory solutions.

7. Conclusion

In this paper, we develop a generic anomaly detection framework for heterogeneous sensor-intensive attributed networks with potentially different types of anomalies that can occur simultaneously across the networks. The framework is generic and can be used for many types of graph structures with no loops. 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. A neural network-based approach is employed to integrate network sensors and topology and transform them into a low-dimensional embedding space. Then, a multilayer perceptron structure is used to predict the status of each node given the sensor data collected at each node and the nodes within each node's propagation set. Finally, two optimization models and important remarks are introduced to find the most likely locations of anomalies and the impacted nodes. The proposed framework can change how sensor data and topological dependencies between network data can be used for anomaly detection in large-scale networks. We will study the temporal effects of anomalies, sensor data, and network topology in future work.

References

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

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

Bhuyan MH, Bhattacharyya DK, Kalita JK (2014) Network anomaly detection: Methods, systems and tools. *IEEE Comm. Surveys Tutorials* 16(1):303–336.

Brice P, Jiang W, Wan G (2011) A cluster-based context-tree model for multivariate data streams with applications to anomaly detection. *INFORMS J. Comput.* 23(3):364–376.

Cui G, Zhou J, Yang C, Liu Z (2020) Adaptive graph encoder for attributed graph embedding. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (ACM, New York), 976–985.

Ding K, Li J, Bhanushali R, Liu H (2019) Deep anomaly detection on attributed networks. *Proc. SIAM Internat. Conf. Data Mining* (Society for Industrial and Applied Mathematics, Philadelphia), 594–602.

Durrett R (2006) *Random Graph Dynamics* (Cambridge University Press, Cambridge, UK).

Feng Z, Fu J, Du D, Li F, Sun S (2017) A new approach of anomaly detection in wireless sensor networks using support vector data description. *Internat. J. Distributed Sensor Networks* 13(1):1–14.

Gao H, Huang H (2018) Deep attributed network embedding. Lang J, ed. *Proc. Twenty-Seventh Internat. Joint Conf. Artificial Intelligence* (AAAI Press, Palo Alto, CA), 3364–3370.

Georgescu S-C (2012) Hbmoa applied to design a water distribution network for a town of 50000 inhabitants. *UPB Scientific Bull. Series D: Mechanical Engrg.* 74(1):91–102.

Giustolisi O, Simone A, Ridolfi L (2017) Network structure classification and features of water distribution systems. *Water Resources Res.* 53(4):3407–3423.

Gogoi P, Bhattacharyya D, Borah B, Kalita J (2011) A survey of outlier detection methods in network anomaly identification. *Comput. J.* 54(4):570–588.

Gulli A, Pal S (2017) *Deep Learning with Kera* (Packt Publishing, Birmingham, UK).

Habeeb R, Nasaruddin F, Gani A, Targio Hashem I, Ahmed E, Imran M (2019) Real-time big data processing for anomaly detection: A survey. *Internat. J. Inform. Management* 45:289–307.

Hamilton WL, Ying R, Leskovec J (2017a) Inductive representation learning on large graphs. Preprint, submitted June 7, https://arxiv.org/abs/1706.02216.

Hamilton WL, Ying R, Leskovec J (2017b) Representation learning on graphs: Methods and applications. Preprint, submitted September 17, https://arxiv.org/abs/1709.05584.

Henderson K, Gallagher B, Li L, Akoglu L, Eliassi-Rad T, Tong H, Faloutsos C (2011) It's who you know: Graph mining using recursive structural features. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (ACM, New York), 663–671.

Kulldorff M (1997) A spatial scan statistic. *Comm. Statist. Theory Methods* 26(6):1481–1496.

Lee S-C, Faloutsos C, Chae D-K, Kim S-W (2017) Fraud detection in comparison-shopping services: Patterns and anomalies in user click behaviors. *IEICE Trans. Inform. Systems* E100.D(10):2659–2663.

Li J, Dani H, Hu X, Liu H (2017) Radar: Residual analysis for anomaly detection in attributed networks. Sierra C, ed. *Internat. Joint Conf. Artificial Intelligence* (International Joint Conferences on Artificial Intelligence, Somerset, NJ), 2152–2158.

Liu S, Qu Q, Wang S (2018) Heterogeneous anomaly detection in social diffusion with discriminative feature discovery. *Inform. Sci.* (439-440):1–18.

Liu F, Ting K, Zhou Z-H (2012) Isolation-based anomaly detection. *ACM Trans. Knowledge Discovery Data* 6(1):1–39.

Liu S, Liu Y, Ni L, Fan J, Li M (2010) Toward mobility-based clustering. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (ACM, New York), 919–927.

Moshtaghi M (2013) Anomaly detection in heterogeneous sensed data. Unpublished PhD thesis, Department of Computing and Information Systems, Melbourne School of Engineering, The University of Melbourne, Melbourne, Australia.

NREL (2016) Bay Area synthetic network. Accessed April 15, 2021, https://egriddata.org/dataset/bay-area-synthetic-network.

Parra L, Sendra S, Lloret J, Bosch I (2015) Development of a conductivity sensor for monitoring groundwater resources to optimize water management in smart city environments. *Sensors (Basel)* 15(9):20990–21015.

Perozzi B, Akoglu L (2016) Scalable anomaly ranking of attributed neighborhoods. *16th SIAM Internat. Conf. Data Mining* (Society for Industrial and Applied Mathematics, Philadelphia), 207–215.

Priebe C, Conroy J, Marchette D, Park Y (2005) Scan statistics on Enron graphs. *Comput. Math. Organ. Theory* 11(3):229–247.

Ranshous S, Shen S, Koutra D, Harenberg S, Faloutsos C, Samatova N (2015) Anomaly detection in dynamic networks: A survey. *Wiley Interdisciplinary Rev. Comput. Stat.* 7(3): 223–247.

Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. *Proc. Twenty-Ninth AAAI Conf. Artificial Intelligence* (AAAI Press, Palo Alto, CA), 4292–4293

Seo Y, Loukas A, Perraudin N (2019) Discriminative structural graph classification. Preprint, submitted May 31, https://arxiv.org/abs/1905.13422.

Speakman I, McFowland E, Neill D (2015) Scalable detection of anomalous patterns with connectivity constraints. *J. Comput. Graphical Statist.* 24(4):1014–1033.

Tango T, Takahashi K (2005) A flexibly shaped spatial scan statistic for detecting clusters. *Internat. J. Health Geographics* 4(11):1–15.

Theissler A (2017) Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. *Knowledge-Based Systems* 123:163–173.

Wang G, Jiao S, Gao X (2009) The application of information fusion technology in the monitoring system of mine pump. *First Internat. Conf. Inform. Sci. Engrg* (IEEE, Piscataway, NJ), 3876–3879.

Yang Q, Barria JA, Green TC (2011) Communication infrastructures for distributed control of power distribution networks. *IEEE Trans. Indust. Inform.* 7(2):316–327.

Yelundur A, Campbell K (2013) Robust parametric empirical Bayes based anomaly detection for flight safety events. *Integrated Comm. Navigation Surveillance Conf.* (IEEE, Piscataway, NJ).