Received May 13, 2022, accepted May 30, 2022, date of publication June 9, 2022, date of current version June 14, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3181742

Recovering Missing Data via Top-*k* Repeated Patterns for Fuzzy-Based Abnormal Node Detection in Sensor Networks

NESRINE BERJAB, HIEU HANH LE, AND HARUO YOKOTA, (Senior Member, IEEE)
Tokyo Institute of Technology, Tokyo 152-8550, Japan

Corresponding author: Nesrine Berjab (berjab@de.cs.titech.ac.jp)

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP 16007.

ABSTRACT The stream data acquired by heterogeneous Internet of Things (IoT) sensors are seldom perfect. Most of the collected data streams include either missing or abnormal values caused by various factors such as failure, malfunction, or integrity attacks. Such unreliable data affect the real-time monitoring and compromise the quality of data analysis. By simply analyzing the sensor data via anomaly detection, applications may still be unreliable over the incomplete sensor data streams. Therefore, a reliable method for recovering the missing data and detecting the abnormal ones is indispensable in the IoT environment. This paper presents FuzHD++, a new method to recover missing sensor data and detect abnormal nodes jointly rather than independently. Both elements, data recovery and abnormal node detection, rely on the observed temporal and spatial correlation of sensor data to effectively achieve reliable recovery estimation and detection performance. In the data recovery process, the system adopts a matrix profile to extract the top-*k* repeated patterns from different sensor nodes. Furthermore, it utilizes the *k*-nearest neighbor estimator to recover the missing data based on the extracted pattern information of multiple neighbor nodes. During the abnormal node detection process, the system adopts a refined fuzzy rule-based detection method. The refined fuzzy rule-based inference system integrates the expert rules and the rules obtained from sensor data analysis to treat the ambiguity in the decision-making process. We validated the performance of FuzHD++ by comparing it with existing methods using two real-world datasets. Our results showed that the proposed missing sensor data recovery method achieves more than 20% improved root mean square error results than most existing methods. Furthermore, FuzHD++ achieved an average accuracy of 92% for analyzing the sensor readings and detecting the abnormal ones. According to the results, the proposed mechanisms based on the observed temporal and spatial correlation analysis improve the robustness of IoT against data loss and integrity attacks.

INDEX TERMS Internet of Things, sensor networks, missing data recovery, abnormal node detection, repeated pattern, fuzzy logic, sensor correlations, false data injection attacks.

I. INTRODUCTION

Since the emergence of the Fourth Industrial Revolution, there has been a growing trend in the use of elements of the Internet of Things (IoT). As a result, IoT environments such as smart homes and smart cities are becoming increasingly popular and have permeated various areas of our lives. In this context, the use of sensors on IoT devices ensures a seamless connection between the devices and the physical world. Indeed, modern IoT devices are computer-like devices that come with a wide range of heterogeneous sensors connected through a dynamic and distributed wireless sensor network (WSN). In this context, the primary requirements of such devices are that they monitor their environmental conditions, report sensor data, and perform appropriate actions in response to the surrounding circumstances [1].

However, the accuracy of such decision-making depends upon the reliability and trustworthiness of the collected sensor data. Unfortunately, the hostile environment of IoT sensors makes this issue more challenging. Most of the stream data

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

obtained from such sensors include either missing or abnormal data caused by various factors such as sensor malfunction, transmission error, storage errors, or malicious attacks. Indeed, the growing popularity of IoT, coupled with physical vulnerability and lack of standardization [1], [2], has led malicious attackers to take an interest in IoT devices. As a result, various types of malicious activities already exist that attempt to compromise the security and privacy of IoT devices. Specifically, some studies [3]–[9] have shown that attackers can compromise and manipulate sensor data in real deployments through false data injection attacks (FDIAs). Such compromised nodes hamper the system's functionality, leading to inappropriate decisions by operators and possibly catastrophic effects [10]. This is where anomaly detection has become a necessity.

To guarantee a safe and reliable IoT system, a myriad of solutions have already emerged that tackle the problem of anomaly detection in sensor data [11], [12], [33]–[40]. However, most of these works do not take into account the presence of missing data within the collected sensor streams. Incomplete sensor data can confuse anomaly detection methods, as they may create false conclusions that lead to wrong detection results. To improve our chances of correctly detecting abnormal sensor nodes, we think it is necessary to ensure the completeness of a sensor data stream before using it in any anomaly detection system. Therefore, it is essential to use appropriate techniques to handle missing and anomalous data with the capability of recovering and detecting them in an integrated framework.

The problem of abnormal node detection was tackled in our previous work [11], in which the spatiotemporal (ST) and multivariate attribute (MVA) correlations of heterogeneous sensor readings were considered in the detection process. The collected sensor data were analyzed through a hierarchical framework based on fuzzy logic to take advantage of domain knowledge and treat the ambiguity in the decision of detecting abnormal nodes. To handle missing data, we carried forward the last observed value. In this context, two major issues remain in the methods presented in the original paper concerning the adopted missing recovering method and the design choice of the fuzzy inference system (FIS). First, the adopted missing recovering method can be described as a hard recovery approach, as we are rigidly forced to pick the single last observed value as the only recovered value. Such a naive interpolation method may not be the best way to handle missing sensor values, and it may affect the abnormal node detection performance. Second, the adopted abnormal node detection method considers a FIS based on background knowledge. However, such a FIS may suffer from a loss of accuracy, especially when dealing with a crafty FDIA.

In this paper, we refer to our prior work as fuzzy-based hierarchical detection (FuzHD), and we tackle its limitations by presenting FuzHD++, a new method to recover missing sensor data and detect abnormal nodes jointly rather than independently. Specifically, in FuzHD++, we propose two new methods, which we refer to as top-*k* repeated patterns (TkRP) and fuzzy-based hierarchical detection with a refined rule base (FuzHD+rRB). These novel methods were devised to handle the missing sensor data and ensure higher abnormal node detection accuracy. In TkRP, we adopt the concept of a matrix profile [13] to extract the top-*k* repeated patterns from different sensor nodes. Furthermore, it utilizes the *k*-nearest neighbor (*k*-NN) estimator to recover the missing data based on the extracted pattern information of multiple neighbor nodes. In FuzHD+rRB, the system adopts a refined fuzzy rule-based detection method to take advantage of domain knowledge and treat ambiguity in the decision-making process. Specifically, we design a hybrid FIS to detect abnormal nodes. Along with the background knowledge-based predefined rules, we also use the so-called Wang–Mendel (WM) method [14], [15] for generating fuzzy rules from sensor data, making it a more comprehensive and flexible FIS.

The main contributions of this paper are as follows.

1) To guarantee an accurate and reliable abnormal node detection, we introduce a new recovery method, TkRP, to correctly recover the missing sensor data.
2) We improve the previously proposed FIS in FuzHD [11] by introducing FuzHD+rRB, in which we design a hybrid FIS to detect abnormal nodes, making it a more comprehensive and flexible FIS.
3) We use a new FDIA threat model to generate a malicious dataset from the original sensor data [12], allowing us to test the abnormal node detection method and evaluate its performance against different threat severity levels.
4) We evaluate our proposed methods through a number of experiments designed to test their parameterization (number of top-k patterns, number of sensor nodes per cluster, length of sensor data streams), accuracy and efficiency.
5) We also augment our evaluation with the Intel Lab dataset [42] in addition to the Yokota Lab dataset, as in our previous work. Our experiments using the two real-world datasets demonstrate that the proposed missing sensor data recovery method TkRP achieves more than 20% improved root mean square error (RMSE) results than most existing methods. Furthermore, the combination of the two proposed methods, FuzHD++, achieves better results than FuzHD in terms of abnormal node detection, with an average accuracy improvement of 14.11%.

The remainder of the paper is organized as follows. Section II reviews the related methods of missing data recovery and anomaly detection in time series. Section III describes some essential background characteristics before introducing our proposed method. Section IV presents the detailed architecture and design of TkRP and FuzHD+rRB to recover missing sensor data and detect abnormal nodes, respectively. We then describe the experimental setup in Section V and present an analysis of the results and evaluation in Section VI. Section VII contains some concluding remarks and perspectives.

II. RELATED WORK

Applications, such as anomaly or abnormal node detection, built upon incomplete sensor data streams are obviously unreliable. If the missing data cannot be filled accurately, existing detection algorithms can hardly be performed. Recovering dirty and missing data could improve clustering over spatial data [16]. For sensor data streams, we argue that recovering the missing values can also improve applications such as abnormal sensor node detection. To guarantee a dependable IoT system, it is essential to conduct studies to deal with these two issues. Although much related research has indeed been carried out, most work tends to focus on recovering missing data or detecting anomalous data, and few studies have simultaneously addressed these two problems.

A. MISSING SENSOR DATA RECOVERY

The straightforward idea is to carry forward the last observed value [11]. However, such a naive interpolation method may not be the best way to handle missing sensor values, as it may confuse the abnormal node detection method and raise false alarms. The estimation algorithms of missing data have been extensively researched by applying different methods; for example, mean impute, k-NN impute, maximum likelihood, Bayes estimator, regression imputation, and delete and multiple imputations [17]. However, none of these methods can be used in sensor data because they can only deal with discrete data and not continuous data. According to the underlying method, recovery algorithms can be classified as either matrix based or pattern based to solve the missing sensor data missing problem.

A matrix-based algorithm transforms the sensor data in a way that allows the application of dimensionality reduction. The singular value decomposition (SVD) method [18] is the most popular method that has been used to achieve such a goal [19]–[23]. Other matrix-based algorithms rely on techniques that differ from SVD, such as principal components analysis [24]–[26], centroid decomposition [27], matrix factorization [28], and nonnegative matrix factorization [29]. All of these matrix-based recovery algorithms multiply back the matrices after reduction and use the results to fill the original missing values. However, the number of reduced dimensions needs to be parameterized as the accuracy–efficiency trade-off is heavily impacted. Moreover, these methods do not consider the sensor spatial correlation. In contrast, pattern-based recovery methods [30]–[32] consider the high spatial and temporal correlation between sensor data streams. When a sensor stream is incomplete with missing values, an algorithm leverages the similarity to any number of reference sensor streams. The observed values in the reference sensor streams are treated as a query pattern. Any incomplete sensor stream matching in that pattern may reveal candidate replacement values in the base streams. Similar to matrix-based algorithms, pattern-based techniques also require predefined user parameters. The length of the query pattern dramatically impacts the accuracy–efficiency trade-off. If the pattern is too small, the technique loses accuracy; if the pattern is too big, the computational time in pattern comparison becomes too costly.

B. ANOMALY DETECTION IN SENSOR DATA

Several techniques for anomaly detection in IoT have been proposed, but most either restrict their application to faults or failures [33]–[35], or to specific network attacks alone [36]–[39]. The area of FDIA detection for WSN has been overlooked by existing works, and only a few studies tackled this issue [11], [12], [40].

In this context, anomaly detection in WSNs can be classified as methods that directly run on sensing devices (i.e., distributed methods) or those running on the cloud (i.e., centralized methods). Performing anomaly detection in a central processing system allows us to adopt complex algorithms and, consequently, to obtain accurate results. A centralized-based approach is proposed where all heterogeneous sensor streams are collected and controlled in a centralized base station [11], [33]. The proposed solution evaluates the intensity of the correlation between the sensor streams by calculating the lag correlation between them.

A centralized failure detection approach is proposed where the base station aggregates the network sensor readings and detects failures by finding an insufficient flow of incoming data [34]. In contrast, distributed methods run directly on sensor nodes equipped with light computation capability. Most of these approaches require historical data samples to be kept in the sensor node, which has limited memory storage. A rule-based distributed fuzzy inference system for WSNs is proposed that combines both local and neighboring observations to identify the occurrence of events [10], [35]. Their experimental results showed that using fuzzy logic improved the accuracy of the event detection. Thus, notwithstanding the limitations of the aforementioned works, few studies have simultaneously addressed both the problem of missing data recovery and abnormal node detection. Instead of simply discarding the sensor streams with missing data, we propose to recover them and then detect the abnormal nodes. Table 1 shows a summary of the characteristics of different approaches along with our proposed method.

In this paper, our proposed methods utilize the observed temporal and spatial correlation of sensor data to achieve reliable estimation and detection performance.

III. PRELIMINARY BACKGROUND

This section provides the essential background characteristics used in our proposed framework and discusses some assumptions about the monitoring environments considered in this paper. A list of abbreviations and notations used in this paper is provided in Table 2 and 3, respectively.

A. SYSTEM AND SENSOR DATA MODEL

An environmental monitoring application in a WSN is defined as an application that monitors the real world and issues a report whenever an event of interest arises during a certain period in a specific location. This paper considers a

TABLE 1. Summary of characteristics of different approaches.

	SVDimp [20], SoftImp [21], SVT [22], Grouse [24], Rosl [25], Spirit [26], CDRec [27], TeNMF [29]	DynaMMo [30], STMVL [31], TKCM [32]	SMART [34]	6thSense [35]	FuzHD [11]	FuzHD++
Missing data recovery	Matrix-based model	Pattern-based model	✗	✗	Partial: last-observed value-based recovery	TkRP: pattern-based model
Anomaly detection	✗	✗	centralized	distributed	distributed	FuzHD+rRB: distributed model
Sensor correlation analysis	✗	Spatiotemporal	Spatial	Temporal	Spatiotemporal and multivariate attributes	Spatiotemporal and multivariate attributes
Sensor networks	Homogeneous	Homogeneous	Heterogeneous	Heterogeneous	Heterogeneous	Heterogeneous

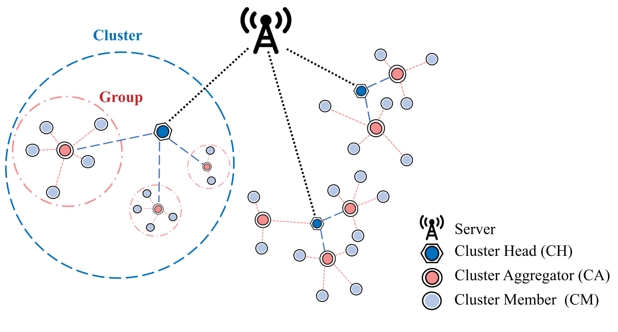


FIGURE 1. A hierarchical distributed wireless sensor network (WSN) based on two-level clustering.

TABLE 2. List of abbreviations and acronyms.

IoT	Internet of things
WSNs	Wireless sensor networks
TkRP	Top-k repeated patterns
FuzHD+rRB	Fuzzy-based hierarchical detection with refined rule base
CH	Cluster head
CA	Cluster aggregator
CM	Cluster member
FDIA	False data injection attack
ST	Spatiotemporal
MVA	Multivariate attribute
MF	Membership function
FIS	Fuzzy inference system
TAS	Temporal average similarity

typical WSN architecture consisting of heterogeneous sensor nodes, a server, and a network connecting all sensor nodes. The server is for collecting and processing sensor data. All the sensor nodes in the WSN are connected to this server directly or indirectly (Figure 1).

This paper addresses the network scalability issue by adopting a hierarchical WSN topology based on two-level clustering. The adopted clustering method allows us to properly utilize the network energy among all nodes, capture the correlation between the sensors, and enhance the system's trustworthiness.

Before describing our proposed approach, we give definitions of the key terms used in this paper. To implement such a thorough monitoring system, n sensor nodes (S_1, $S_2, \ldots, S_n$) are geographically divided into clusters, each covering a certain area. Each cluster should include one cluster head (CH) and other heterogeneous cluster member (CM) nodes arranged into groups according to their type. Each group is controlled by a cluster aggregator (CA). The CMs are responsible for sensing and collecting various attributes, such as temperature, humidity, and light intensity. The CA is responsible for all communication between the CM and CH nodes. Once all the sensed data within the cluster are collected, the CH forwards the messages directly to the server. Figure 1 depicts the topology of this hierarchical network formed by two-level clustering.

TABLE 3. Symbols and notations.

Symbol	Notation
S_i	Sensor i
$T(S_i)$	Type of sensor i
$L(S_i)$	Geographical location of sensor i
$C(S_i)$	Cluster of sensor i
$size(C(S_i))$	Size of the cluster Cp where sensor i belongs
$\overrightarrow{O(S_i)}$	Data stream of sensor i
$O(S_i, t)$	Measurement of sensor i at time t
$\overrightarrow{O(S_i)}_{t,o}$	Subsequence of sensor i of length o and starting from t

Each sensor node (i.e., CM, CA, and CH) has a unique identifier, where $i \in [1, n]$. Each S_i is characterized by six attributes, L, T, C, A, input I, and output O.

Definition 1: Let $L(S_i)$ be the *location of sensor* S_i, specified by its geographic coordinates x_i, y_i, and z_i.

Definition 2: Let D be the set of *sensor types*, where D includes *Temperature*, *Humidity*, *Light*, and *Smoke*.

Definition 3: Let $T(S_i)$ be the node's *sensor type*, where $T(S_i) \in D$.

Definition 4: We denote *size*(C) as the number of sensors deployed in the *cluster* C. Let $C(S_i)$ be the cluster within which S_i is located. The clustering formation is based on a defined distance threshold, th_d. Two sensors, S_i and S_j, belong to the same cluster C if and only if $C(S_i) = C(S_j)$, and the distance between $L(S_i)$ and $L(S_j)$ is less than th_d.

Definition 5: In addition to the clustering, homogeneous sensor nodes within the same cluster are divided into *groups* according to their type $T(S_i)$. Let $A(S_i)$ denote the group within which S_i is located. The two sensor nodes S_i and S_j belong to the same group within the same cluster C if and only if $T(S_i) = T(S_j)$ and $C(S_i) = C(S_j)$.

Definition 6: Let $I(S_i, S_j, t)$ denote an *input* report message received by S_i from S_j at time t.

Definition 7: Finally, let $\overrightarrow{O(S_i)}$ be S_i's *sensor data stream*, where $\overrightarrow{O(S_i)} = \{O(S_i, 1), \ldots, O(S_i, t), \ldots, O(S_i, m)\}$. $O(S_i, t)$ is the *sensor node's output* data stream with every S_i sensing data at time t, and m is the length of the sensor data stream.

B. ASSUMPTIONS

Our research is based on the following assumptions.

- To reduce the complexity of the problem, we assume that every sensing environment is characterized by its environmental conditions, such as temperature, light intensity, and relative humidity.
- As noted, the clustering concept is adopted for the network topology. Although several complex and innovative clustering techniques have been proposed for WSNs, this paper considers a very simple clustering technique for environmental monitoring in WSNs.
- All clusters should be composed of homogeneous and heterogeneous sensor nodes to maintain high event-detection accuracy.
- Depending on the application, a CH node can be a special sensor with more potential than other sensor nodes in terms of energy, bandwidth, and memory. However, in this paper, we consider that all the sensor nodes in the network have the same performance characteristics. In addition, the role of the CH is periodically rotated among all nodes to balance the energy consumption and the traffic load in the network.
- In this paper, the CAs or the CHs do not aggregate the collected data. Instead, we need to keep the actual collected data from each sensor to recover the missing data and detect the abnormal data.
- N-modular redundancy is used to achieve a dependable and fault-tolerant WSN. Furthermore, the considered WSN must satisfy a good distribution of the clusters where at least three sensor nodes must be deployed within one cluster (i.e., triple modular redundancy is a particular case of N-modular redundancy).
- While some sensor nodes may be compromised and considered abnormal nodes, we assume that the majority of the sensors will remain trustworthy.

IV. PROPOSED APPROACH

Our proposed approach aims to guarantee the system's dependability by recovering the missing sensor data and detecting abnormal nodes. This problem can be expressed as follows.

Problem: Given n coevolving correlated sensor stream sequences provided by n heterogeneous sensors collected at the same time, recover the missing sensor data, determine at any point in time which sensors are abnormal, and report all such nodes.

To address these challenges and guarantee reliable and secure monitoring of WSN, we propose two new methods, called TkRP and FuzHD+rRB, for detecting abnormal nodes in a heterogeneous WSN while recovering the missing sensor data. Both TkRP and FuzHD+rRB utilize the observed temporal and spatial correlation of sensor data to achieve reliable estimation and detection performance. First, the proposed TkRP adopts a matrix profile to extract the top-k repeated patterns from different sensor nodes and utilize k-NN pattern information of neighbor nodes to recover the missing data. Second, it detects abnormal nodes using a fuzzy logic-based hierarchical detection method. The proposed framework is depicted in Figure 2, which shows the various sensor node modules and the flowchart for recovering missing data and processing abnormal nodes. We follow a three-step process to achieve our objective, as described in detail in the following subsections.

A. STEP 1 IN FuzHD++: DATA ACQUISITION AND FuzHD+rRB ABNORMAL NODE-LOCAL DETECTION

The first step involves collecting heterogeneous sensor streams from the various clusters deployed in the monitored area and performing the abnormal node-local detection. In the following subsections, we explain the details related to the local detection module in FuzHD+rRB.

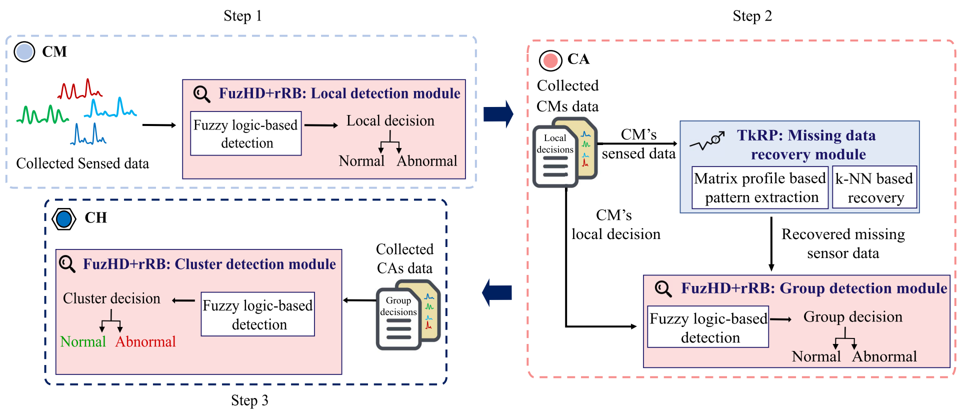
1) DEFINITION OF THE INPUT/OUTPUT VARIABLES ALONG WITH THEIR MEMBERSHIP FUNCTION

The CM senses environmental events and executes the local detection process to check whether the newly collected data are subject to abnormality. Figure 3 illustrates details of the design of the adopted scheme for the local detection module. This detection module considers temporal semantic correlations to derive a crisp local decision. Every CM maintains a short-term history of the collected sensed data. This aggregation of data is used to construct a sliding time window containing the most recent sensed data in the sensor node stream. In the literature on stream processing, time windows are a familiar concept [33].

In this paper, we use the time window not only as a mechanism for bounding the sensor node stream aggregation but also to profile the behavior of the sensor node readings over time [11]. The sensed data will be time correlated, and the variation range will usually be small in the short term [41]. By using the time window concept, we can derive valuable information regarding the sensor nodes' temporal similarity. The sensed data contain a k-second timestamp, indicating the time at which the sensor node reported the reading.

As shown in Figure 4, the sensor node time-series samples are grouped into $(p + 1)$ frames to compose the sliding time window of size W_l, where $l \in \{0, p\}$.

As time passes, the window slides in one-frame increments over the sensor data stream. Each frame contains T



Step 1
CM
Collected Sensed data
FuzHD+rRB: Local detection module
Fuzzy logic-based detection
Local decision
Normal
Abnormal
Step 2
CA
Collected CMs data
Local decisions
CM's sensed data
TkRP: Missing data recovery module
Matrix profile based pattern extraction
k-NN based recovery
CM's local decision
Recovered missing sensor data
FuzHD+rRB: Group detection module
Fuzzy logic-based detection
Group decision
Normal
Abnormal
CH
FuzHD+rRB: Cluster detection module
Cluster decision
Fuzzy logic-based detection
Collected CAs data
Group decisions
Normal
Abnormal
Step 3

FIGURE 2. Overview of FuzHD++: Fuzzy-based hierarchical abnormal node detection with top-*k* repeated patterns as a missing data recovery method.

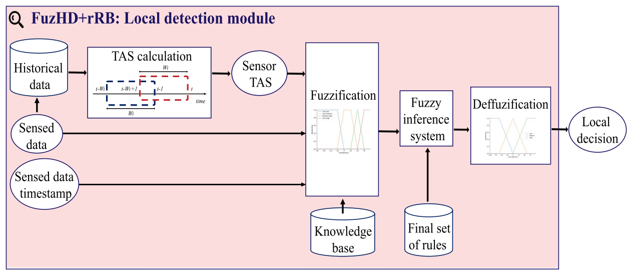
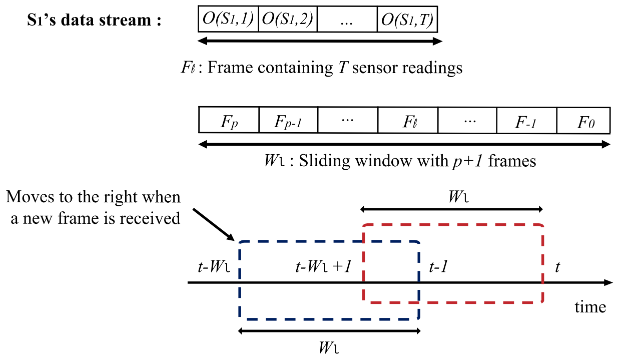


FIGURE 3. FuzHD+rRB: Local detection scheme for CMs.



S1's data stream :
O(S1,1)
O(S1,2)
...
O(S1,T)
Fl : Frame containing T sensor readings
Fp
Fp-1
...
Fl
...
F-1
F0
Wl : Sliding window with p+1 frames
Moves to the right when a new frame is received
Wl
t-Wl
t-Wl +1
t-1
t
time
Wl

FIGURE 4. Temporal average similarity (TAS) based on a sliding window.

successive sensor readings. After setting the sliding time window, we apply a summarization function to extract the relevant information about sensor node temporal similarity. F_0 is the frame containing the recently collected sensor readings. F_1 is the frame for the T previous sensor readings. $p+1$ is the size of the sliding time window, and T is the number of sensor readings within each frame. For each frame F_l within the window, we calculate the temporal similarity between the frame F_l and the current frame F_0 [11]. The temporal similarity is given by Equation (1).

$$q(F_l, F_0) = \frac{1}{(1 + \sqrt{\sum_{t=1}^{T}(O(S_i,t)_{F_l} - O(S_i,t)_{F_0})^2}))} \tag{1}$$

The temporal average similarity (TAS) between the current frame data and the data in the window is then calculated. As indicated in Equation (2), the average similarity is calculated by adding a weighted summation to the calculation. The closer the frame is to the current timestamp frame, the more it is correlated [11].

$$Q(F_0) = \frac{\sum_{l=1}^{p} weight \cdot q(F_l, F_0)}{l+1}$$
$$\text{where the weight: } weight = \frac{1}{T_{F_0} - T_{F_l}} \tag{2}$$

Here, T_{f_0} is the last timestamp of the collected sensor reading $O(S_i, T)$ in frame f_0. The same applies to T_{f_l}, where it is the last timestamp of the collected sensor reading $O(S_i, T)$ in frame f_l. The smaller the TAS, the more the frame at the current timestamp deviates from the historical sensor node data. After the CM finishes the calculation of the TAS, it conducts the fuzzy local detection process. Indeed, this paper proposes a fuzzified methodology for detecting abnormal nodes in the network. It uses fuzzy logic to identify the severity of abnormality of sensor nodes rather than just giving crisp results. Moreover, a fuzzy-based system tends to provide rules that are by nature easy to interpret.

Thereby, together with the obtained TAS value, both the current raw sensed value $O(S_i, t)$ and its timestamp are fuzzified through predefined membership functions (MFs). Choosing to include the sensed data timestamp is highly significant for the accuracy of the detection [42]. The monitored environment can differ for each time-of-day segment. As a result, the input–output response will also differ depending on the time of day. For example, the light intensity and temperature during the day are generally higher than at night.

In this paper, we consider an environment monitored by sensor nodes for temperature, humidity, light, and smoke density. For each type of sensor node, the local detection module takes three linguistic variables as its input: sensed value, average temporal similarity, and sensed data timestamp. In the fuzzification process, the three crisp values are converted into degrees of membership, with each membership using the trapezoid and triangle models. The trapezoidal function is determined as follows:

$$trapezoid(x; a, b, c, d) = \begin{cases} 0 & \text{if } x < a \\ \dfrac{x-a}{b-a} & \text{if } a \le x \le b \\ 1 & \text{if } b \le x \le c \\ \dfrac{d-x}{d-c} & \text{if } c \le x \le d \\ 0 & \text{if } d > x \end{cases} \quad (3)$$

where x is a member of the universal set, and the parameters a, b, c, d (with $a < b < c < d$) determine the x coordinates of the four corners of the underlying trapezoidal MF. As for the triangular MF, the function is specified by three parameters $\{a, b, c\}$ as follows:

$$triangular(x; a, b, c) = \begin{cases} 0 & \text{if } x \le a \\ \dfrac{x-a}{b-a} & \text{if } a \le x \le b \\ \dfrac{c-x}{c-b} & \text{if } b \le x \le c \\ 0 & \text{if } c \ge x \end{cases} \quad (4)$$

The sensed value input is one of temperature *Te*, humidity *Hu*, light *Li*, or smoke *Sm*. The trapezoidal and triangular MF for the *Te* variable has four semantic values, i.e., very low *VL*, low-to-medium *LM*, medium-to-high *MH*, and very high *VH*. For instance, the MFs for temperature are shown in Figure 5. The triangular MFs for the *Hu*, *Li*, and *Sm* variables have three semantic values, i.e., low *Lo*, medium *Me*, and high *Hi*. The triangular MF for average temporal similarity *TAS* has two semantic values, i.e., small *Sm* and big *Bi*. Finally, the trapezoidal MF for the sensed data timestamp *TS* has four semantic values, i.e., night *Nt*, morning *Mo*, afternoon *Af*, and evening *Ev* [11]. After being fuzzified, the three fuzzy inputs are then fed into the fuzzy inference system.

2) GENERATION OF THE REFINED RULE SET FOR FuzHD+rRB

The inference engine of the proposed FIS uses the Mamdani-type fuzzy process. FIS can be designed either from domain knowledge or from data. In FuzHD [11], the adopted FIS is based on expert knowledge. However, it may suffer from a loss of accuracy under different environmental conditions. Rule-based legibility is essential to take full advantage of FIS. For this reason, here we improve the previously proposed FIS in our FuzHD by designing a hybrid FIS that cooperates between two kinds of information, namely background knowledge and hidden knowledge in data. In FuzHD+rRB, the generation of the fuzzy rule base for the FIS is conducted offline and is decomposed into two main phases (Figure 6).

- In the rule induction, we use the WM method for generating fuzzy rules from the sensor sample data along with the background knowledge-based predefined rules.

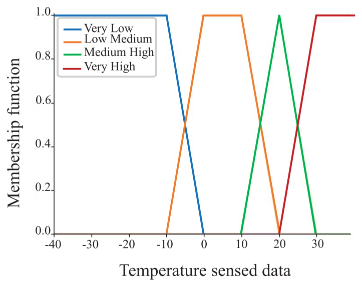


FIGURE 5. Membership functions (MFs) for the input linguistic variables: Temperature sensed value.

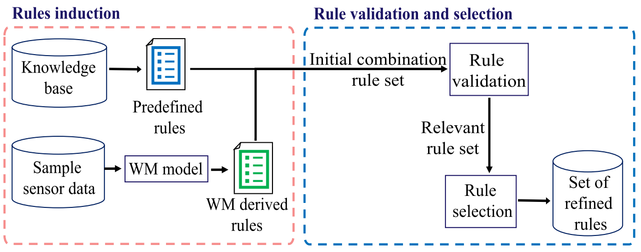


FIGURE 6. Generation of the refined rule set for FuzHD+rRB.

- In the rule validation and selection, the initial combination rule set is further checked to only keep those that are relevant and select those that are appropriate for inclusion in the refined rule set.

a: RULE INDUCTION

First, we start by acquiring the predefined rules. The form of these rules is as follows:

IF premise, THEN consequent

where the premise is the fuzzy input variables connected by *and* and *or* logical connectors, and the consequent is the fuzzy output variable. More formally:

If x_1 is A_1^i and x_2 is A_2^i …and x_p is A_p^i then y is Ci.

The fuzzy sets A_j^i are those for which the MF of x_j^i is maximum for each input variable j from pair i. The fuzzy set C_i is that for which the MF of the observed output, y_i, is maximum.

The predefined fuzzy rule base comprises a set of rules designed to decide the probability of the node being abnormal. By considering the background knowledge, we use heuristics to build the rule base for our abnormal node detection. An example might be as follows.

IF *Te* is *Hi* AND *TAS* is *Sm* AND *TS* is *Ni*, THEN *Abnormal* is *Hi*.

For instance, the predefined fuzzy rules related to temperature sensor nodes are listed in Table 4. This set of rules contains the rules involving linguistic variables based on the sensed values from temperature sensor nodes. The rule base for the other sensor types can be constructed similarly [11].

Once we acquire the predefined rules, we move to the WM-derived rules. The adopted WM model follows the following three steps.

1) Each variable of the input space is automatically divided into fuzzy regions. The WM model does not impose any

TABLE 4. Predefined rule base for temperature sensor nodes.

Rule ID	Temperature Te	TAS	Timestamp TS	Abnormal
1	Very Low	Big	Afternoon	High
2	Very Low	Big	Morning	Low
3	Very Low	Big	Night	Low
4	Very Low	Big	Evening	High
5	Very Low	Small	Morning	Medium
6	Very Low	Small	Afternoon	High
7	Very Low	Small	Night	Medium
8	Very Low	Small	Evening	High
9	Low Medium	Big	Morning	Low
10	Low Medium	Big	Night	Low
11	Low Medium	Big	Afternoon	Low
12	Low Medium	Big	Evening	Low
13	Low Medium	Small	Night	Medium
14	Low Medium	Small	Morning	Medium
15	Low Medium	Small	Evening	Medium
16	Low Medium	Small	Afternoon	High
17	Medium High	Big	Afternoon	Low
18	Medium High	Big	Morning	Low
19	Medium High	Big	Night	Medium
20	Medium High	Big	Evening	Medium
21	Medium High	Small	Night	High
22	Medium High	Small	Afternoon	Medium
23	Medium High	Small	Night	High
24	Medium High	Small	Evening	High
25	High	Big	Morning	High
26	High	Big	Night	High
27	High	Big	Evening	Medium
28	High	Big	Afternoon	Low
29	High	Small	Morning	High
30	High	Small	Night	High
31	High	Small	Evening	Medium
32	High	Small	Afternoon	Medium

specific partition for the input variables. Indeed, they are equally partitioned on a predefined number of triangular membership fuzzy sets. Each domain interval is divided into 2N+1 regions. The center of each MF lies in the center of the region, and the extreme lies at the center of the next region.

2) Then, we generate the fuzzy rules from the given data pairs from the sample sensor data. One fuzzy rule is generated for each input–output data pair from the sample data. The output is computed through centroid defuzzification.
3) Finally, the conflicting rules are removed. For example, the rules that share the same antecedent but with different consequents are removed.

After acquiring both the predefined and WM-derived rules, we merge the two sets to obtain the initial combination rule set.

b: RULE VALIDATION AND SELECTION

In the rule validation, we aim to identify the relevant rule set from the initial combination rule set. Clearly, certain rules will show some redundancy after combining the two sets. Therefore, such redundant rules need to be removed and not included in the relevant rule set. Moreover, rules having a number of MF fuzzy sets that differ from that defined by the WM model are also removed and are not included in the relevant rule set. In the rule selection, we aim to select the appropriate rules to be included in the refined rule set. At this stage, we assess the conflicting rules and rank them according to their prediction accuracy in the sample sensor data. A rule is added to the refined rule set if the expected predictive accuracy of the rule meets the desired accuracy and is not subsumed by a conflicting rule with a lower expected predictive accuracy.

3) DEFUZZIFICATION

Finally, the defuzzification process of obtaining a single number from the output of the aggregated fuzzy set is conducted. It is used to transfer FIS results into a crisp output, which is used to make a local decision and send a report message to the CH for further analysis that eventually leads to the final decision. The centroid method is used to calculate the crisp value of the fuzzy output as follows:

$$x^* = \frac{\sum_{i=1}^{k} x_i \mu(x_i)}{\sum_{i=1}^{k} \mu(x_i)} \tag{5}$$

where x^* is the crisp value, x_i represents each member of the output universe, $\mu(x_i)$ is the aggregated output MF, and k is the number of items in the fuzzy set.

The confidence for abnormal node detection is defined as the output. The triangular MF for the fuzzy output variable is defined in terms of three levels, i.e., low *Lo*, medium *Me*, or high *Hi*. Figure 7 shows the MFs for abnormal detection confidence. The linguistic variables represent the detector's confidence about the presence of a data integrity attack. For example, if the local detection value is higher than 30, we are more than 30% certain that it is an abnormal node. If the detector's confidence is smaller than 30%, it is more than likely that it is not an abnormal node.

B. STEP 2 IN FuzHD++: TkRP MISSING DATA RECOVERY AND FuzHD+rRB ABNORMAL GROUP DETECTION

After collecting all the data (i.e., sensed data and local decisions) from the CMs, the CA verifies whether there are missing values within the sensed values. When missing values occur in the CMs, the CA executes the missing data recovery. In this subsection, we introduce our new recovery method that uses the matrix profile (Figure 8) to extract the top-*k* repeated patterns from sensor nodes within the same group and utilize the *k*-NN pattern information of neighbor nodes to recover the missing data. The top-*k* repeated patterns are only extracted from the CMs identified as a normal sensor during the local detection process.

1) TkRP: CONSTRUCTION OF THE REFERENCE PATTERN DATABASE

The first step toward recovering the missing sensor data is constructing the reference pattern database. Figure 9 illustrates the construction process. From each collected sensor data stream $\overrightarrow{O(S_i)}$, the CA starts computing the stream's matrix profile.

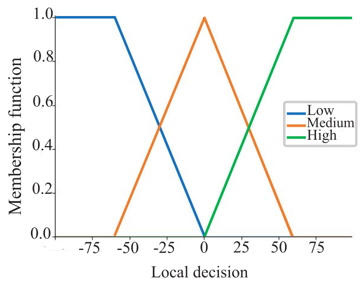


FIGURE 7. Abnormal node confidence MF.

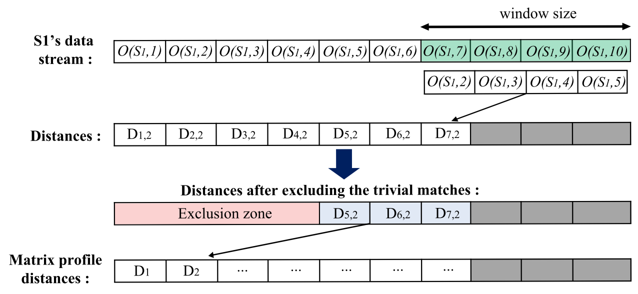

window size
S1's data stream :
Distances :
Distances after excluding the trivial matches :
Exclusion zone
Matrix profile distances :

FIGURE 8. Example of matrix profile calculation.

The matrix profile is a recently proposed data structure [12] that annotates a time series to solve the problem of anomaly detection and motif discovery.

Besides its novelty, the method is robust, scalable, and parameter free. Hence, we adopt this data structure for our proposed recovery method. The matrix profile comprises two primary components, namely, a distance profile and a profile index. The distance profile is a vector of minimum Z-normalized Euclidean distances. The profile index contains the index of its first nearest neighbor, i.e., it is the location of its most similar subsequence.

Definition 8: A *subsequence* $\overrightarrow{O(S_i)}_{t,o}$ of a sensor S_i's data stream $\overrightarrow{O(S_i)}$ is a continuous subset of the values from $\overrightarrow{O(S_i)}$ of length o starting from t. Formally, $\overrightarrow{O(S_i)}_{t,o} = \{O(S_i, t), O(S_i, t+1), \ldots, O(S_i, t+o-1)\}$, where $1 \leq t \leq o - m + 1$.

The algorithms that compute the matrix profile use a sliding window approach. Given a sensor data stream $\overrightarrow{O(S_i)}$ and the desired subsequence $\overrightarrow{O(S_i)}_{t,o}$ of length o, first, we use a sliding window of length o to extract all subsequences of length o.

Definition 9: An *all-subsequences* set B of a sensor S_i's data stream $\overrightarrow{O(S_i)}$ is an ordered set of all the possible subsequences of $\overrightarrow{O(S_i)}$ obtained by a sliding window of length o across $\overrightarrow{O(S_i)}$: $B = \left\{\overrightarrow{O(S_i)}_{1,o}, \overrightarrow{O(S_i)}_{2,o}, \ldots, \overrightarrow{O(S_i)}_{m-o+1}\right\}$.

Then, we compute the pairwise distance among these windowed subsequences against the entire sensor data stream $\overrightarrow{O(S_i)}$. The distance calculations occur $m - o + 1$ times, where m is the length of the sensor data stream and o is the window size. Because the subsequences are pulled from the sensor data stream itself, an exclusion zone is required to prevent trivial matches. For example, a subsequence of sensor data stream matching itself or a subsequence of sensor data stream very close to itself is considered a trivial match. The exclusion zone is simply half of the window size before and after the current window index. The values at these indices are ignored when computing the minimum distance and the nearest neighbor index. Figure 8 illustrates an example of matrix profile calculation. It shows the computation of a distance profile starting at the second window. The matrix profile stores the distances in Euclidean space, meaning that a distance close to 0 is most similar to another subsequence in the sensor data stream, and a distance far away from 0, say 100, is unlike any other subsequence.

With the matrix profile computed, it becomes simple to find the top-k of repeated patterns [43].

Definition 10: The *most repeated pattern* is a pair of subsequences $\left\{\overrightarrow{O(S_i)}_{v,o}, \overrightarrow{O(S_i)}_{w,o}\right\}$ of a sensor S_i's data stream $\overrightarrow{O(S_i)}$ that are most similar. More formally, $\forall a, b, v, w$ the pair $\left\{\overrightarrow{O(S_i)}_{v,o}, \overrightarrow{O(S_i)}_{w,o}\right\}$ is the subsequence pattern if and only if $distance(\overrightarrow{O(S_i)}_{v,o}, \overrightarrow{O(S_i)}_{w,o}) \leq distance(\overrightarrow{O(S_i)}_{a,o}, \overrightarrow{O(S_i)}_{b,o})$; where $|v - w| \geq gap_{th}$ and $|a - b| \geq gap_{th}$ for gap_{th}: the gap that exists between the subsequences where $gap_{th} < 0$.

Definition 11: The k^{th} *most repeated pattern* is the k^{th} most similar pair $\left\{\overrightarrow{O(S_i)}_{v,o}, \overrightarrow{O(S_i)}_{w,o}\right\}$.

Here for simplicity, we only deal with pairs. However, we can also extend the notion of most repeated patterns to a set of subsequences that are very similar to each other.

Once we finish extracting the top-k repeated patterns, we extract the snippets. For each extracted pair of subsequences within the top-k repeated patterns, we extract their snippet.

Definition 12: The *most repeated pattern snippet* is a subsequence of the most repeated pattern.

Time series snippets are defined to describe the most representative subsequences in a time series [44]. The primary use of snippets here is to find the subsequence patterns that occupy most of each top-k repeated pattern in question to summarize the repeated pattern at a high level. In this paper, instead of extracting the k_{th} snippet [39], we are only interested in extracting the top-1 snippet for each extracted pair of subsequences within the top-k repeated patterns. Indeed, the top-1 snippet is undoubtedly the most representative pattern that summarizes the subsequence. For each extracted snippet, we check if a similar one already exists in the reference pattern dataset, and if not, insert it.

Definition 13: The *reference pattern dataset* is a database that contains a set of d snippets that represent the most repeated patterns of sensors located within the same group.

Having a reference database of patterns, we are now able to recover the missing sensor data.

2) TkRP: MISSING DATA RECOVERY METHOD

Figure 10 depicts the workflow of the missing recovery method. The CA collects the latest observations generated by the CMs' sensor data stream. A null value is included in the stream when missing observations are reported. If missing measurements are reported, the CA requests an estimation of missing values. This estimation is based on the k-NN algorithm and uses the previously constructed reference database that includes the top-k repeated patterns. It checks if a similar experience already exists in the reference dataset and identifies patterns similar to the current one. After identifying similar patterns, the CA computes the k-NN distances and generates the estimated values.

Then, the recovered sensor data stream follows the process described in subsection B.1 to check if the stream includes any new patterns and updates the reference pattern database when it is necessary.

3) ABNORMAL NODE GROUP DETECTION

The group detection module operates at the CA level by considering spatial semantic correlations. In this detection module, a more accurate decision is made by including the local decisions of multiple homogeneous CMs located in the same group within the same cluster. After receiving a local decision message from a CM, the CA stores the crisp decision value. After collecting all the CMs' local decisions, the CH executes the cluster detection process for each CM node to give a group decision about the node's abnormality. The group detection module uses two inputs, the CMs' crisp local decisions and the CMs' local decisions. The fuzzifier converts the crisp values into degrees of membership by applying the corresponding MF. After being fuzzified, a sigma-count factor [45] is used as a measure of fuzzy cardinality to quantify the CMs' local decisions.

$$\sum Count(f) = \sum \mu f(S_i) \tag{6}$$

Here, f is a fuzzy set characterized by an MF $\mu f(S_i)$, which gives the degree of similarity for S, and $S_i = (S_1, S_2, \ldots, S_n)$ is the set of CMs. Precisely, $f(S_i)$ is the property of interest related to the sensor node's local decision, e.g., "Abnormal level is high." A fuzzy majority quantifier is then used to obtain a fuzzified indication of the consensual CMs' local decisions. For a more accurate decision, we use the *Most* quantifier to characterize the fuzzy majority of the CMs' local decisions, which takes any value from the interval 0 to 1 as the truth value of its proposition [46].

$$u_{most}\left(\frac{\sum Count(f)}{|S_i|}\right) = u_{most}\left(\frac{\sum_i \mu(S_i)}{n}\right) \tag{7}$$

Next, the fuzzified inputs and the quantified CMs' local decisions are fed into the fuzzy inference process. The fuzzy rule base comprises a set of rules designed to decide about the abnormality of the CM. An example of the format of the rule is "IF *Abnormal* is *H* AND *Most(CMsDecision)* is *L* THEN *Abnormal* is *H*." Fuzzy inference combines the rules to obtain an aggregated fuzzy output. Finally, the defuzzifier converts the fuzzy output variable back to a crisp value that is used to make a group decision and reported to the CH.

TABLE 5. Experiment environment.

	Raspberry Pi 2	Raspberry Pi 3
CPU	ARM Cortex-A7 Quad Core 900 MHz	ARM Cortex-A53 Quad Core 1.2 GHz
RAM	1 GB	1 GB
OS	Debian 9.3 (Raspbian Stretch)	Debian 9.3 (Raspbian Stretch)
Python	2.7.13	2.7.13

TABLE 6. Experimental parameters.

Parameters	Value
Normal temperature	[15°C, 30°C]
Normal humidity	[30%, 50%]
Normal smoke density	[0 ppm, 0.2 ppm%]
Normal light intensity	[0 Lux, 2000 Lux]
New sensor readings	every 2 min
F_l	5 sensor readings
W_l	5 frames
Number of nodes per cluster	$\{4, 5, 6\}$
top-k	10
$u_{most}(x)$	0 if $x \leq 0.3$ $2x - 0.6$ if $0.3 < x < 0.8$ if $x \geq 0.8$

C. STEP 3 IN FuzHD++: FuzHD+rRB ABNORMAL NODE CLUSTER DETECTION

Cluster identification is processed at the CH level by considering the ST and MVA sensor correlations. In this detection module, a more accurate decision is made by including the group decisions of multiple heterogeneous CAs located in the same cluster. After receiving a group-decision message from a CA, the CH stores the crisp decision value. After collecting all the CA group decisions, the CH performs the fuzzy inference for each sensor node to give the cluster decision about the node's abnormality. The detection mechanism is similar to that for group detection. However, compared with the group decision, the cluster decision considers the observations from heterogeneous sensor nodes in addition to only homogeneous sensor nodes.

The CH's fuzzy rule base comprises a set of rules designed to determine the CM's abnormality. An example of the rule might be "IF *Abnormal* is *Lo* AND *Most(CAsDecision)* is *Lo*, THEN *Abnormal* is *Lo*." If abnormal nodes are detected, the CH sends a report message to the server.

V. EXPERIMENTAL SETUP

This section describes the datasets used to evaluate our proposed approach and the details of the sensor network that we

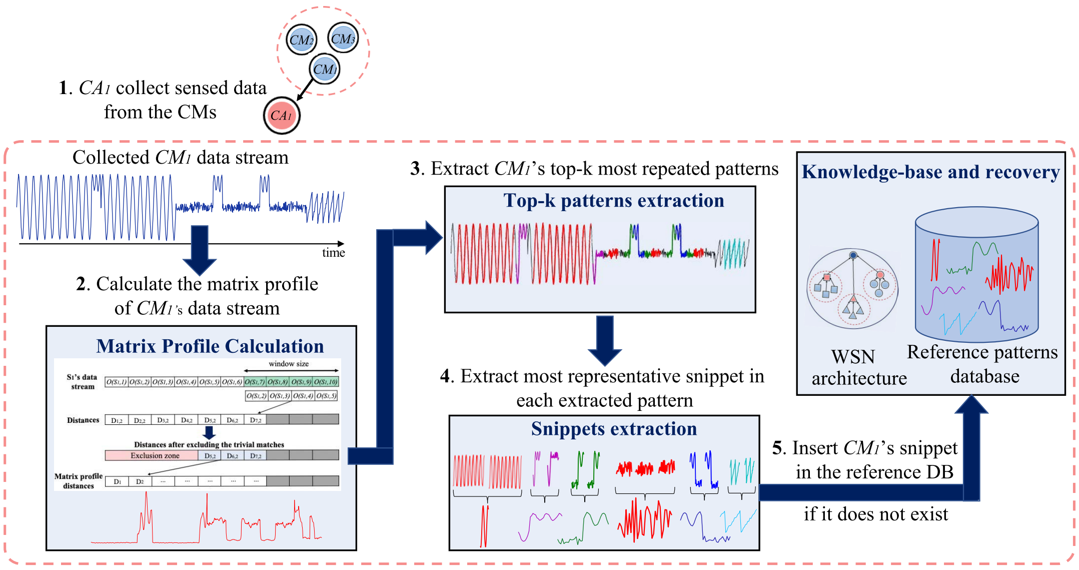

1. CA1 collect sensed data from the CMs
CM2
CM3
CM1
CA1
Collected CM1 data stream
time
2. Calculate the matrix profile of CM1's data stream
Matrix Profile Calculation
3. Extract CM1's top-k most repeated patterns
Top-k patterns extraction
4. Extract most representative snippet in each extracted pattern
Snippets extraction
Knowledge-base and recovery
WSN architecture
Reference patterns database
5. Insert CM1's snippet in the reference DB
if it does not exist

FIGURE 9. Missing data recovery process in TkRP.

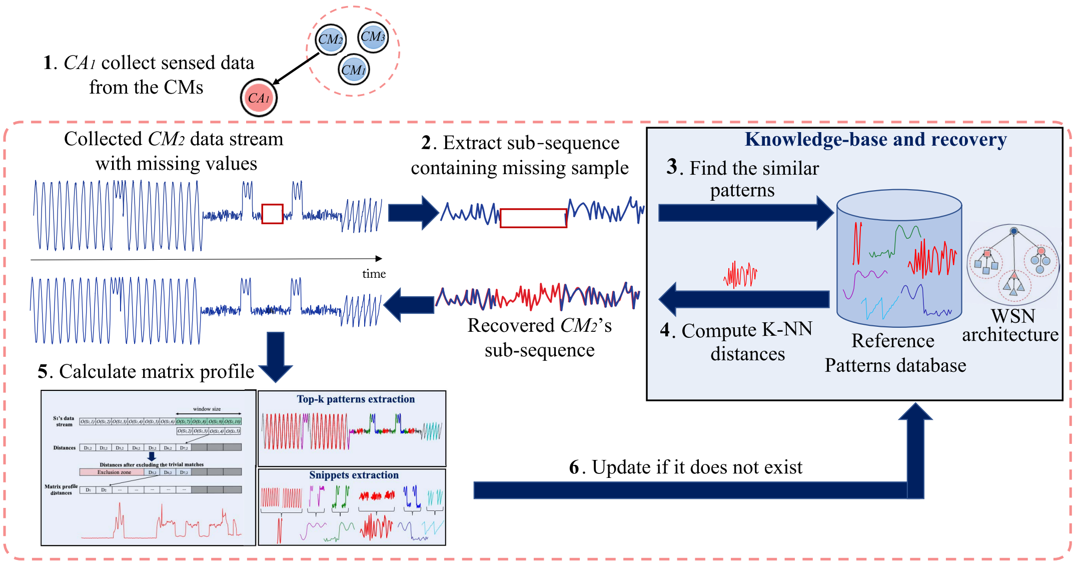

1. CA1 collect sensed data from the CMs
CM2
CM3
CM1
CA1
Collected CM2 data stream with missing values
time
2. Extract sub-sequence containing missing sample
Knowledge-base and recovery
3. Find the similar patterns
4. Compute K-NN distances
Reference Patterns database
WSN architecture
Recovered CM2's sub-sequence
5. Calculate matrix profile
Top-k patterns extraction
Snippets extraction
6. Update if it does not exist

FIGURE 10. Missing data recovery process in TkRP.

have implemented, including the deployment setting and the experimental scenario design.

A. DATA ACQUISITION

To show that our proposed approach applies to real-world WSNs deployed with heterogeneous sensors, we use two datasets that have different types of sensor deployment. The first dataset is the publicly available Intel Berkeley Research Lab dataset [47]; the second is the Yokota Lab where the data are collected from our deployable WSN in our laboratory.

1) INTEL BERKELEY RESEARCH LAB DATASET

In this dataset, the data were collected from 54 sensor nodes deployed in the Intel Berkeley Research Lab between February 28 and April 5, 2004. To effectively monitor the whole lab environment, 54 sensors are unevenly distributed in different locations in the research lab. Mica2Dot sensors with weatherboards are used to collect time-stamped topology information, along with temperature (in degrees Celsius), humidity (temperature-corrected relative humidity ranging from 0 to 100%), light (Lux) (a value of 1 Lux corresponds to moonlight, 400 Lux to a bright office, and 100,00 Lux to full sunlight), and voltage values (in volts ranging from 2 to 3). The batteries, in this case, were lithium-ion cells that maintained a reasonably constant voltage over their lifetime. A new reading was collected every 31 seconds. In total, 2.3 million readings were collected from these sensors. The sensors were dispersed in the lab, as shown in Figure 11.

2) YOKOTA LAB DATASET

In addition to the Intel Lab dataset, we also collected sensor data streams from 27 sensor nodes in our laboratory between January 24 and July 25, 2018. The real-world sensor data were collected periodically while performing our usual daily activities. The sensor nodes were deployed using the Raspberry Pi 2 and 3 Model B microcontroller platforms (Table 5), as we consider the Raspberry Pi to be the best IoT hardware platform in terms of performance and flexibility. Each physical sensor node is equipped with one temperature sensor module (DS18B20 temperature sensor), one humidity sensor (DHT11 humidity sensor), one smoke density sensor (MQ-2 smoke sensor), and one digital light intensity sensor (BH1750 digital light sensor), yielding a total of 64 sensors. The technical characteristics of the Raspberry Pi platforms, sensors, and server used in our experimental setting are described in Tables 4 and 5. As shown in Figure 12, the sensor nodes were divided into five clusters separated from each other and with different environmental conditions. Two clusters comprised five sensor nodes each and were located in our laboratory room. The third cluster consisted of six physical nodes located in a kitchen corner and exposed to sunlight, the fourth consisted of six physical nodes located in a seminar room, and the fifth consisted of five physical nodes located in a server room. Each sensor transmits data approximately every two minutes, giving a total of 20.9 million readings.

B. DATA PREPROCESSING

Three main steps must be performed to prepare the dataset for evaluation: cleaning the raw sensor data, injecting false sensor data, and physically separating the sensor nodes into clusters. Cleaning the data is necessary to ensure that the proposed abnormal node detection is only executed on known FDIAs, allowing for consistent evaluation. After that, new false sensor data may be injected. The clustering process is also considered a necessary process here to capture the sensor data correlation adequately. In the following subsections, we explain the three main steps in more detail.

1) DATA CLEANING

To use the Intel Berkeley Research Lab dataset, we encountered an issue related to the notion of time variation (i.e., epoch) within the collected data. Indeed, the usage of the epoch is necessary to build a baseline that works on sensor data streams such as our collected dataset or the Intel Lab dataset.

However, for the case of the Intel Lab dataset and even our dataset, the notion of the epoch is loosely defined. Indeed, even though sensor nodes are commanded to collect a new reading in every defined epoch, the fact of having multiple values or missing values for different epochs cannot be avoided (because of failures or communication problems). For the WSNs deployed at the Intel Lab, the reasons behind the failures were communication problems and the sensor battery condition. In addition, we found that readings of

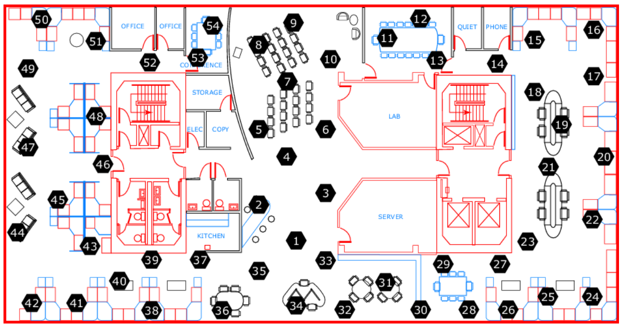


FIGURE 11. Sensors deployed in the Intel Berkeley Research Lab.

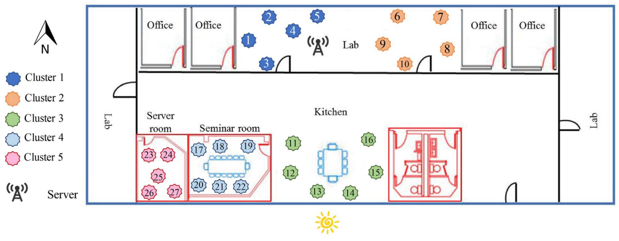


FIGURE 12. Heterogeneous sensor nodes deployed in the Yokota Lab.

sensor node five in the Intel Lab data were not recorded. Consequently, it was removed from the dataset.

Concerning our deployed WSNs, some sensor nodes had missing data for different epochs because of SD card corruption. However, because of the sensor constraints, we found that the epoch was not strongly defined in either dataset. Indeed, both datasets were missing less than 10% of the expected measurements. Thus, we needed to standardize the concept of epoch and set it to a well-defined size. To unify the size, we split the readings into epochs of two minutes each. The proposed missing sensor data recovery TkRP was used to substitute each missing sensed value in an epoch. If a sensor had more than one reading during the epoch, we took the average of these measurements. We recovered missing data using the original dataset containing normal sensor readings. In other words, at this stage, we are still not dealing with malicious data. The summary of the datasets is shown in Table 7.

2) FALSE SENSOR DATA INJECTION

Given the lack of sensor datasets with malicious data for WSNs and the need to test the accuracy of our approach, we propose an FDIA model to create an attack strategy. An attacker's goal in the context of FDIA is to evoke or hide events without triggering the detection alarm. The primary challenge is to maintain a balance between the outcome of the attack and the risk of being detected. In our prior work [11], the proposed attack models were unsophisticated and not comprehensive enough to support the claims in the paper. In that work, we only considered three trivial cases where the attacker deliberately either randomly changes a sensor reading or selects the minimum or the maximum

TABLE 7. Datasets.

	Number of sensors	Sensor types	Number of clusters	Period	Dataset size	Missing rate
Yokota Lab	27	Temperature, humidity, light intensity, and smoke	5 (each includes 5 or 6 sensor nodes)	Between January 24 and July 25, 2018	20.9 million readings (new reading every 2 min)	9.28%
Intel Lab	54	Temperature, humidity, and light intensity	11 (each includes 4 or 5 sensor nodes)	Between February 28 and April 5, 2004	2.3 million readings (new reading every 2 min)	7.78% (without including sensor 5)

possible value. We carefully chose the type of attacked sensor, insertion time, and attack period. With such a proposed attack strategy, it is impossible to guarantee a variety of attack patterns, resulting in uninteresting attack outcomes that are easy to detect. In contrast, WSNs are subject to various threats where we cannot simply anticipate the attacker's actual intention.

To tackle this issue, here we use an attack strategy that generates a malicious dataset from the original sensor data, allowing us to test the detection algorithm and evaluate its performance against different threat severity levels [12]. We create evaluation data, including FDIA based on the initially collected dataset. The occurrence probability of malicious data depends on the exponential distribution:

$$f(e) = \frac{1}{\epsilon} exp(-\frac{e}{\epsilon}) \tag{8}$$

where $(500 \leq \epsilon \leq 1000)$. In addition, we defined nine types of false data. The difference in the false injected data between the real data and the evaluation data depends on a Gaussian distribution:

$$f(e) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{(e - O(S_i, t))^2}{2\sigma^2}\right\}, \tag{9}$$

where $\sigma \in [1, 2, 3, \cdots 9]$. We referred to both Equations (5) and (6) and injected false sensor data readings into the initially collected sensor data. The sensor type, FDIA type, and insertion time were chosen randomly. With such an FDIA strategy, the attack can be very stealthy and deceive the detection mechanism without being easily detected.

VI. EVALUATION

Each conducted experiment was repeated twice, and the average results were taken.

A. EVALUATION OF TkRP IN TERMS OF ACCURACY

To measure TkRP accuracy, as our evaluation metric, we adopt the most commonly used measure, that is, RMSE between the original value and the recovered value:

$$RMSE = \sqrt{\frac{1}{|Tm| \sum_{t \in Tm}(O(S_i, t) - \overline{O(S_i, t)})^2}} \tag{10}$$

where Tm is the set of missing values, $O(S_i, t)$ is the original value, and $\overline{O(S_i, t)}$ is the recovered value.

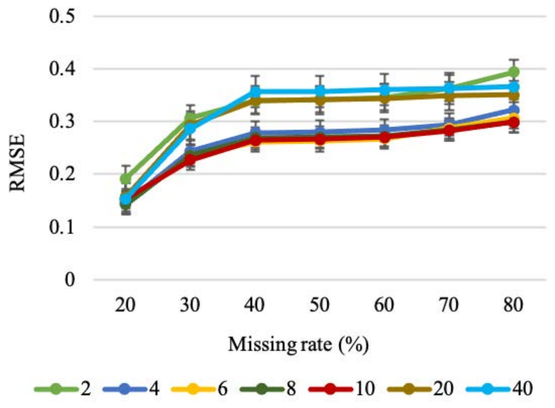


FIGURE 13. Accuracy of TkRP with an increasing number of top-*k* repeated patterns: Intel Lab dataset.

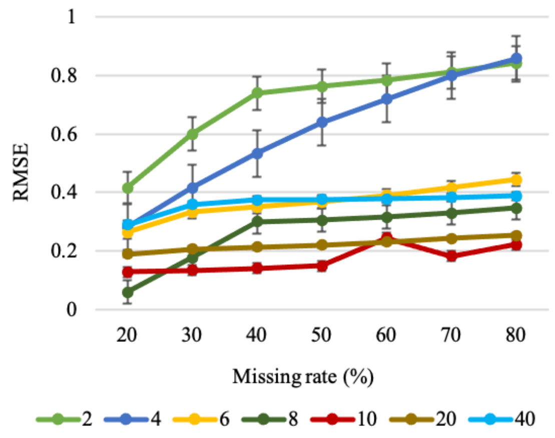


FIGURE 14. Accuracy of TkRP with an increasing number of top-*k* repeated patterns: Yokota Lab dataset.

To compare the efficiency and accuracy of TkRP against state-of-the-art recovery algorithms (introduced in Table 1), we use the recent benchmark that evaluates missing value recovery techniques in time series [48]. We set missing sensor values to appear arbitrarily in the middle of a randomly chosen sensor data stream in the dataset. We then vary the size of the missing values from 20% to 80% (of the chosen sensor data stream) and measure the average recovery accuracy using RMSE. We normalize the error across all algorithms using a z-score (the lower, the better) and present the results in Figures 13–21.

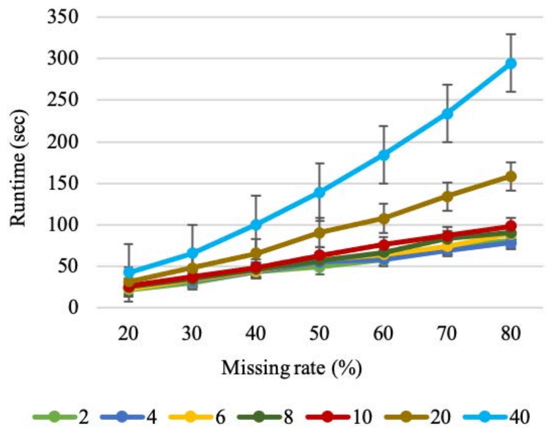


FIGURE 15. Runtime of TkRP with an increasing number of top-*k* repeated patterns: Intel Lab dataset.

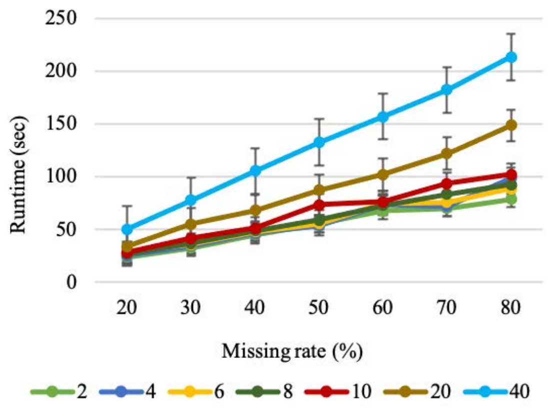


FIGURE 16. Runtime of TkRP with an increasing number of top-*k* repeated patterns: Yokota Lab dataset.

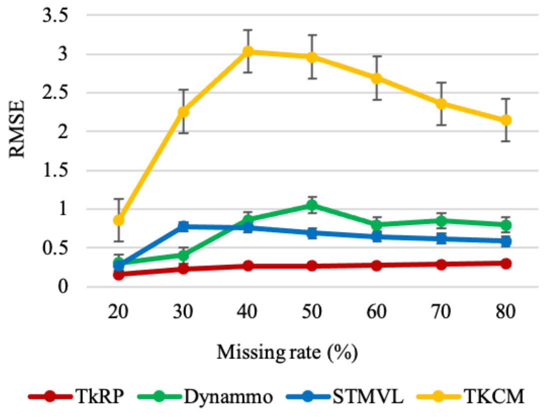


FIGURE 17. Accuracy comparison between pattern-based methods: Intel Lab dataset.

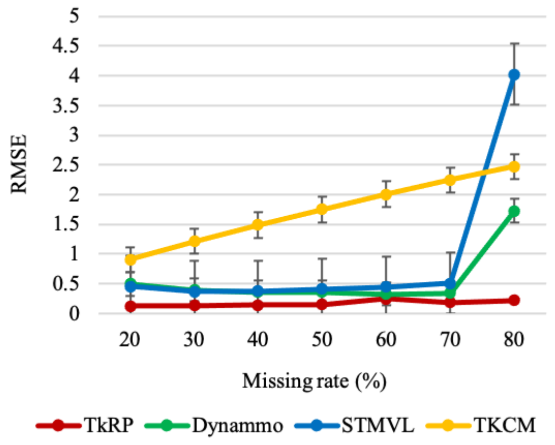


FIGURE 18. Accuracy comparison between pattern-based methods: Yokota Lab dataset.

1) EVALUATION OF TkRP WITH AN INCREASING NUMBER OF TOP-*k* PATTERNS

In this set of experiments, we used the Intel Lab dataset and the Yokota Lab subdataset (one-month period). The most critical parameter for TkRP is the number of top-*k* extracted repeated patterns. We observed that the runtime of TkRP increases along with the increased value, k (Figures 15 and 16). This result was expected because the higher the value of k, the higher the number of comparisons performed to produce the recovery (causing more time- and space-intensive computations). Surprisingly, increasing k did not always improve accuracy (Figures 13 and 14). The pattern extraction used by TkRP keeps only the most repeated patterns and filters out the rest.

At some point, the extra extracted patterns resort to infrequent pattern extraction that corrupts the recovery. To achieve a suitable trade-off between accuracy and efficiency, the best top-*k* proved to be $k \in \{8, 9, 10\}$.

2) COMPARISON OF TkRP WITH OTHER RELATED WORKS

In this set of experiments, we study the effect of traffic load associated with the decrease in the amount of sensed data on the performance of TkRP. In other words, we evaluate the recovery accuracy when increasing the percentage of missing values in one sensor data stream. For the evaluation, we use the Intel Lab dataset and the Yokota Lab sub-dataset (i.e., one-month period). We also present the accuracy results using RMSE. We set the size of the top-*k* to 10. Figures 17–21 show the results. The results show that TkRP outperforms the other algorithms. Indeed, this experiment shows that, in general, TkRP takes advantage of having correlated sensor streams to produce better accuracy. The improvement is more noticeable in the case of the Yokota Lab dataset, although both datasets stand out by their very high correlation between the sensor streams. However, in the Yokota Lab dataset, the correlation between the sensors is higher than in the Intel Lab dataset because of the small distance between the CMs. This is why TkRP, which captures such correlations by design, performs so well compared with the other related works. We also observe in this experiment that the error does not always increase along with the size of the missing block. However, as shown in Figure 19, the runtime of TkRP increases almost linearly as the missing rates increase. Thus, although the runtime of TkRP is slower than in the pattern-based methods, from a practical point of view, it is still reasonable.

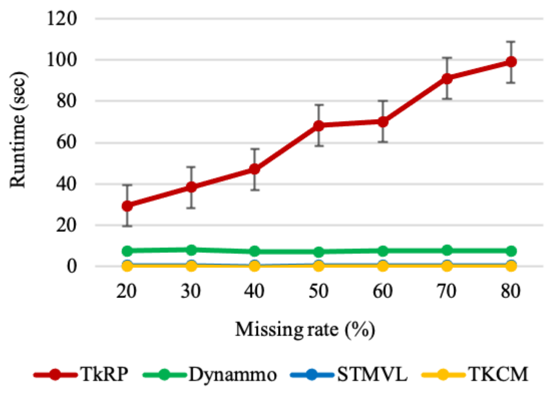


FIGURE 19. Runtime comparison between pattern-based methods: Yokota Lab dataset.

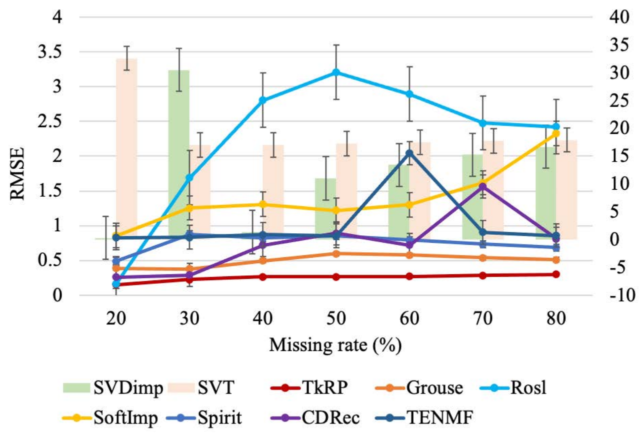


FIGURE 20. Accuracy comparison between matrix-based methods: Intel Lab dataset.

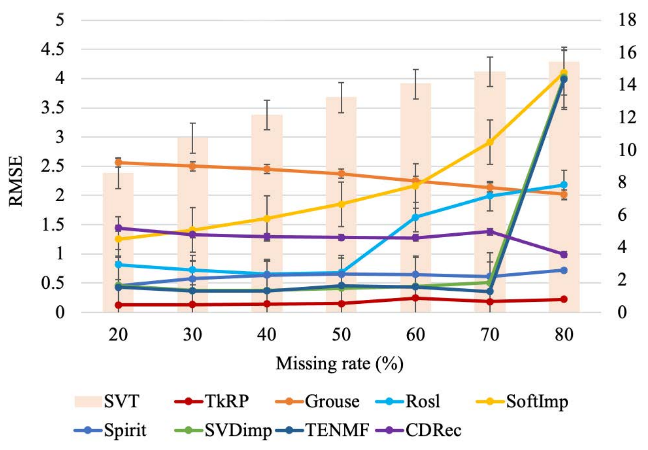


FIGURE 21. Accuracy comparison between matrix-based methods: Yokota Lab dataset.

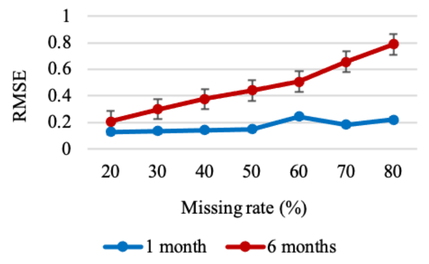


FIGURE 22. Accuracy with a large dataset: Yokota Lab for a period of six months.

3) SCALABILITY WITH A LARGE DATASET

In this experiment, we study the effect of traffic load associated with the increase in the amount of sensed data on the performance of TkRP. In other words, we evaluate the scalability of TkRP when we are dealing with an increase in the length of sensor data streams. For the evaluation, we use the Yokota Lab sub-dataset (i.e., one-month period) and the whole Yokota Lab dataset (i.e., six-month period). Figure 22 illustrates the experiment results. The increase in RMSE occurs as expected because adding more incomplete sensor streams increases the number of missing values. However, the accuracy of the results is still under 1.

4) IMPACT OF THE NUMBER OF SENSOR NODES PER CLUSTER ON THE PERFORMANCE OF TkRP

In this set of experiments, we evaluate the impact of the number of sensor nodes per cluster on TkRP performance. Figure 23 depicts the efficiency of TkRP and recovery accuracy on the Intel lab dataset when increasing the number of sensor nodes per cluster. We only use the Intel lab dataset for these experiments since it has more sensor nodes scattered in different areas. We set the size of the missing block to 20%. The sensor stream length is fixed when the number of sensor nodes per cluster varies. Here, we also use the average RMSE and runtime values to assess the efficacity and efficiency of TkRP. The experiment shows that the RMSE accuracy of TkRP remains largely unaffected when we vary the number of sensor nodes per cluster. This was expected, as TkRP is only interested in extracting the top-k repeated patterns from neighboring sensor nodes and disregards the rest. It can be seen from the results that when the number of neighbor nodes is small (less than 4), the selected neighbor nodes may not be in the adjacent area of the node's physical location. Thus, errors may happen while recovering the actual values of the incomplete sensor streams. Nevertheless, there is a slight decrease in RMSE when the number of sensors increases from 5 to 11. That is due to the increase in the number of nodes sharing similar physical locations.

B. ACCURACY OF THE ABNORMAL NODE DETECTION METHOD

To evaluate the combination of the two proposed methods FuzHD++ (FuzHD+rRB with TkRP) in terms of abnormal node detection, four performance metrics were used: accuracy, precision, recall, and F1-score. We used precision, recall, and F1-score to quantify the detection accuracy. Accuracy is the degree to which the detection results confirm the actual values. Precision is the percentage of the actual abnormal nodes among the identified nodes. Recall is the percentage of the identified abnormal nodes among the actual

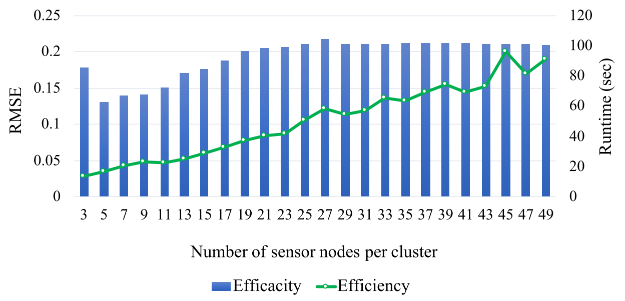


FIGURE 23. Impact of the number of sensor nodes per cluster on the performance of TkRP: Intel Lab dataset.

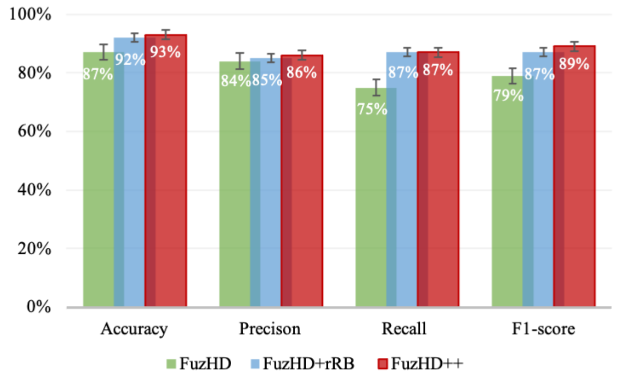


FIGURE 24. FuzHD++ for better abnormal node detection accuracy: Yokota Lab dataset.

abnormal sensors. The F1-score is a measure of a test's accuracy and is calculated from the precision and recall of the test. In addition, we used one month of data from the Yokota Lab dataset and one week of data from the Intel Lab dataset as the sample sensor data to generate the WM-derived rules.

Figures 23 and 24 show the evaluation results of the two datasets measuring the extent to which the combination of the two proposed methods (i.e., FuzHD++ and the previous method FuzHD) detects abnormal nodes.

Even though the environmental conditions for each cluster in the two datasets differed, the proposed combined method, FuzHD++, achieved good detection accuracy with a low false-positive rate for the task of analyzing the sensor readings to determine whether the sensor nodes were behaving normally or had been exposed to FDIAs.

The results show that FuzHD++ achieved an average accuracy of 92%, average precision of 84%, recall rate of 85.50%, and average F1-score of 85%. Moreover, FuzHD++ achieved better detection results than those achieved using FuzHD, with an average accuracy improvement of over 14.11%, improved average precision of 8.13% and recall rate of 14%, and a higher average F1-score of 11.05%. Therefore, we conclude that our proposed method detects abnormal nodes with high accuracy.

Furthermore, Figures 23 and 24 show the evaluation results of FuzHD++ and FuzHD+rRB (with the last observed values as the missing recovery method) using the two datasets. The results show that by adopting the proposed TkRP method

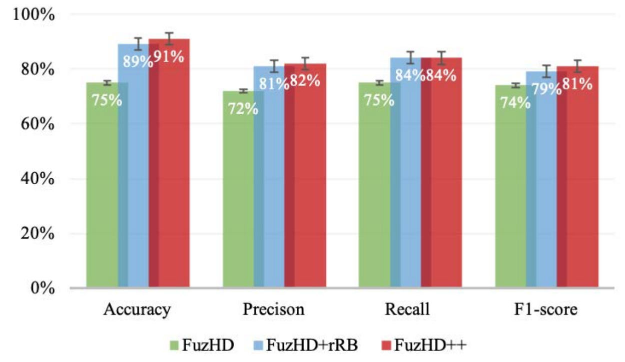


FIGURE 25. FuzHD++ for better abnormal node detection accuracy: Intel Lab dataset.

as a missing recovery method, the abnormal node detection accuracy achieved an average accuracy improvement of over 1.6%. Overall, both datasets did not suffer from high missing rates (less than 10%), explaining the low improvement.

1) EVALUATION OF FuzHD++ UNDER HIGHER MISSING DATA RATES

To investigate this issue further, we broke down the Yokota Lab dataset into smaller defined groups. Table 8 illustrates the breakdown of the missing data rate of the Yokota Lab dataset by one-month period and cluster location. From the obtained results, we selected three cases with high missing data rates as case studies to evaluate the importance of FuzHD++ for better detection accuracy. The first case evaluates the abnormal node detection accuracy of the cluster located in the kitchen area during the period from March 24 to April 24 (with a medium missing data rate of 25.44%). The second case evaluates the abnormal node detection accuracy of the cluster located in the seminar room during the period from June 24 to July 25 (with a medium–high missing data rate of 35.08%). Finally, the third case evaluates the abnormal node detection accuracy of the cluster located in the seminar room during the period from March 24 to April 24 (with a high missing data rate of 58.73%). Figure 25 illustrates the F1-score evaluation of FuzHD, FuzHD+rRB with the last observed values as a missing recovery method, and FuzHD++ for each case study. The results show that FuzHD++ achieved the best detection results with an average F1-score improvement of over 8.22%. Thus, instead of simply replacing the missing values with the last observed values, the proposed recovery method TkRP is a better choice for FuzHD+rRB to achieve higher abnormal node detection accuracy.

2) IMPACT OF THE NUMBER OF SENSOR NODES PER CLUSTER ON THE PERFORMANCE OF FuzHD++

In this set of experiments, we evaluate the impact of the number of sensor nodes per cluster on FuzHD++ performance. Figure 27 depicts the detection accuracy and the efficiency of FuzHD++ on the Intel lab dataset when increasing the number of sensor nodes per cluster. Here, we use the average

TABLE 8. Breakdown of the missing data rate of the Yokota Lab dataset by one-month periods and cluster location.

	Jan–Feb	Feb–Mar	Mar–Apr	Apr–May	May–June	June–July
Seminar room	7.89%	32.50%	**58.73%**	0.19%	18.80%	**35.08%**
Lab	5.61%	7.99%	1.95%	0.08%	0.08%	0.11%
Kitchen	4.34%	10.94%	**25.44%**	0.09%	1.79%	5.67%
Server room	0.52%	1.72%	2.08%	0.08%	0.10%	0.18%

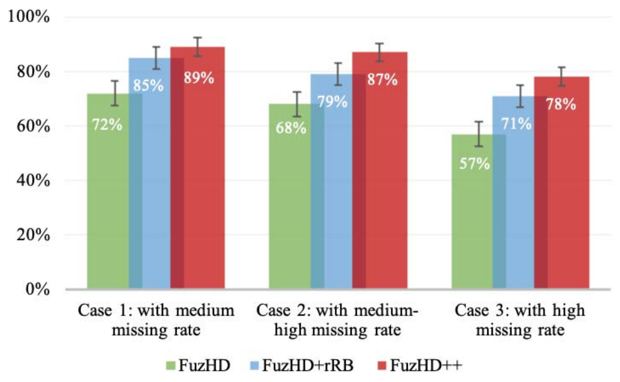


FIGURE 26. F1-score evaluation of FuzHD++ under higher missing data rates: Yokota Lab dataset.

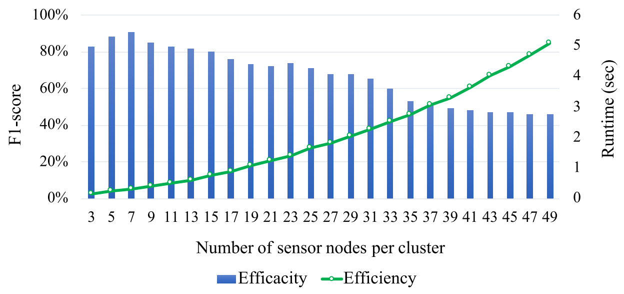


FIGURE 27. Impact of the number of sensor nodes per cluster on the performance of FuzHD++: Intel Lab dataset.

F1-score and runtime values to assess the abnormal node detection accuracy and efficiency of FuzHD++. It can be seen from the results that when the number of neighbor nodes is small, the selected neighbor nodes may not be in the neighboring area of the node's physical location, which may cause a wrong decision. Indeed, FuzHD++ uses the temporal and spatial correlation of the sensing data between neighboring nodes to detect the abnormal nodes and assist the fuzzy logic-based decision-making system. With the continuous increase of the number, the number of nodes sharing similar spatial environment conditions gradually increases, and the accuracy of the detection rate is also continuously improved. However, with many nodes grouped into one cluster, FuzHD++ will use the sensed data of nodes with a longer physical distance to participate in the decision-making, which will lead to an increase in the false detection rate. Thus, FuzHD++ depends not only on the number of neighboring sensor nodes to achieve a higher detection rate but also on the distance or the spatial correlation between intra-cluster neighboring sensors.

VII. CONCLUSION

To improve the chances of correctly detecting abnormal nodes in the sensor network, we think it is necessary to ensure the completeness of a sensor data stream before using it in any anomaly detection system. This paper presents FuzHD++, a new method that handles missing and anomalous data with the capability of recovering and detecting them in an integrated framework. In FuzHD++, the two new methods, TkRP and FuzHD+rRB, are devised to recover missing data and detect abnormal nodes, respectively. The observed temporal and spatial correlations of sensor data are utilized in both elements to effectively achieve reliable estimation and detection performance. In TkRP, we adopt a matrix profile to extract the top-k repeated patterns from different sensor nodes. Furthermore, it utilizes the k-NN estimator to recover the missing data based on the extracted pattern information of multiple neighbor nodes. In FuzHD+rRB, we adopt a refined fuzzy rule-based abnormal node detection method with a refined rule base. The refined rule-based FIS integrates the expert rules and the rules obtained from sensor data analysis, making it a more comprehensive and flexible inference system. We demonstrated the effectiveness of our proposed methods by conducting a variety of performance evaluations. We evaluate our proposed methods through a number of experiments designed to test their parameterization (number of top-k patterns, number of sensor nodes per cluster, length of sensor data streams), accuracy, and efficiency. Our experiments using two real-world datasets demonstrate that the proposed missing sensor data recovery method TkRP achieves improved RMSE results of over 20% compared with most existing methods. Furthermore, the combination of the two proposed methods, FuzHD++, achieves better results than FuzHD in terms of abnormal node detection, with an average accuracy improvement of over 14.11%. Besides, the experiment results show that the proposed method depends not only on the number of neighboring sensor nodes but also on the distance or the spatial correlation between intra-cluster neighboring sensors to achieve a higher detection rate.

FuzHD++ can detect abnormal sensor nodes under FDIAs but cannot detect other types of attacks at this moment. As for future work, we are currently working on this part. Such expansion may contribute to enhancing the overall IoT security issues.

REFERENCES

[1] L. Mottola and G. P. Picco, "Programming wireless sensor networks: Fundamental concepts and state of the art," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 1–51, Apr. 2011.

[2] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, ''Sensor network data fault types,'' *ACM Trans. Sensor Netw.*, vol. 5, no. 3, pp. 1–29, May 2009.

[3] D. Padmavathi and M. Shanmugapriya, ''A survey of attacks, security mechanisms and challenges in wireless sensor networks,'' *Int. J. Comput. Sci. Inf. Secur.*, vol. 4, nos. 1–2, pp. 1–9, 2009.

[4] C. Franzen. (2013). *Dick Cheney had the Wireless Disabled on his Pacemaker to Avoid Risk of Terrorist Tampering*. Accessed: Apr. 7, 2022. [Online]. Available: https://www.theverge.com/2013/10/21/4863872/dick-cheney-pacemaker-wireless-disabled-2007

[5] T. Kavitha and D. Sridharan, ''Security vulnerabilities in wireless sensor networks: A survey,'' *J. Inf. Assurance Secur.*, vol. 5, no. 1, pp. 31–44, 2010.

[6] N. Berjab, C. M. Yu, S. Y. Kuo, and H. Yokota, ''Impact analysis for DoS and integrity attacks on IoT systems,'' in *Proc. 7th Int. Conf. Inf. Syst. Technol.*, 2017, pp. 1–8.

[7] A.-Y. Lu and G.-H. Yang, ''False data injection attacks against state estimation in the presence of sensor failures,'' *Inf. Sci.*, vol. 508, pp. 92–104, Jan. 2020.

[8] J. Radcliffe. (2011). *Hacking Medical Devices for Fun and Insulin: Breaking the Human SCADA System*. Accessed: Apr. 7, 2022. [Online]. Available: https://media.blackhat.com/bh-us-11/Radcliffe/BH_US_11_Radcliffe_Hacking_Medical_Devices_WP.pdf

[9] B. D. Weinberg, G. R. Milne, Y. G. Andonova, and F. M. Hajjat, ''Internet of Things: Convenience vs. privacy and secrecy,'' *Bus. Horizons*, vol. 58, no. 6, pp. 615–624, Nov. 2015.

[10] G. Liang, S. R. Weller, J. Zhao, F. Luo, and Z. Y. Dong, ''The 2015 Ukraine blackout: Implications for false data injection attacks,'' *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3317–3318, Jul. 2017.

[11] N. Berjab, H. H. Le, C.-M. Yu, S.-Y. Kuo, and H. Yokota, ''Hierarchical abnormal-node detection using fuzzy logic for ECA rule-based wireless sensor networks,'' in *Proc. IEEE 23rd Pacific Rim Int. Symp. Depend. Comput. (PRDC)*, Dec. 2018, pp. 289–298.

[12] N. Berjab, H. H. Le, and H. Yokota, ''A spatiotemporal and multivariate attribute correlation extraction scheme for detecting abnormal nodes in WSNs,'' *IEEE Access*, vol. 9, pp. 135266–135284, 2021.

[13] C.-C.-M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, ''Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets,'' in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1317–1322.

[14] L.-X. Wang and J. M. Mendel, ''Generating fuzzy rules by learning from examples,'' *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.

[15] L. Wang, ''The WM method completed: A flexible fuzzy system approach to data mining,'' *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 6, pp. 768–782, Dec. 2003.

[16] S. Song, C. Li, and X. Zhang, ''Turn waste into wealth: On simultaneous clustering and cleaning over dirty data,'' in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1115–1124.

[17] L. Gruenwald, H. Chok, and M. Aboukhamis, ''Using data mining to estimate missing sensor data,'' in *Proc. 7th IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Oct. 2007, pp. 207–212.

[18] D. Skillicorn, *Understanding Complex Datasets: Data Mining With Matrix Decompositions* (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series). Boca Raton, FL, USA: CRC Press, 2007.

[19] J. He, Y. Zhou, G. Sun, and T. Geng, ''Multi-attribute data recovery in wireless sensor networks with joint sparsity and low-rank constraints based on tensor completion,'' *IEEE Access*, vol. 7, pp. 135220–135230, 2019.

[20] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, ''Missing value estimation methods for DNA microarrays,'' *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[21] R. Mazumder, T. Hastie, and R. Tibshirani, ''Spectral regularization algorithms for learning large incomplete matrices,'' *J. Mach. Learn. Res.*, vol. 11, pp. 2287–2322, Jan. 2010.

[22] J.-F. Cai, E. J. Candés, and Z. Shen, ''A singular value thresholding algorithm for matrix completion,'' *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, Mar. 2008.

[23] J. He, G. Sun, Y. Zhang, and T. Geng, ''Data recovery in heterogeneous wireless sensor networks based on low-rank tensors,'' in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2016, pp. 616–620.

[24] L. Balzano, Y. Chi, and Y. M. Lu, ''Streaming PCA and subspace tracking: The missing data case,'' *Proc. IEEE*, vol. 106, no. 8, pp. 1293–1310, Aug. 2018.

[25] X. Shu, F. Porikli, and N. Ahuja, ''Robust orthonormal subspace learning: Efficient recovery of corrupted low-rank matrices,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 3874–3881.

[26] S. Papadimitriou, J. Sun, C. Faloutsos, and P. S. Yu, ''Dimensionality reduction and filtering on time series sensor streams,'' in *Managing and mining sensor data*. Springer: Boston, MA, USA, 2013, pp. 101–141.

[27] M. Khayati, M. Böhlen, and J. Gamper, ''Memory-efficient centroid decomposition for long time series,'' in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar. 2014, pp. 100–111.

[28] H. Yu, N. Rao, and I. S. Dhillon, ''Temporal regularized matrix factorization for high-dimensional time series prediction,'' in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 847–855.

[29] J. Mei, Y. de Castro, Y. Goude, and G. Hébrail, ''Nonnegative matrix factorization for time series recovery from a few temporal aggregates,'' in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2382–2390.

[30] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos, ''DynaMMo: Mining and summarization of coevolving sequences with missing values,'' in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 507–516.

[31] X. Yi, Y. Zheng, J. Zhang, and T. Li, ''ST-MVL: Filling missing values in geo-sensory time series data,'' in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2016, pp. 2704–2710.

[32] K. Wellenzohn, M. H. Böhlen, A. Dignös, J. Gamper, and H. Mitterer, ''Continuous imputation of missing values in streams of pattern-determining time series,'' in *Proc. 20th Int. Conf. Extending Database Technol. (EDBT)*, 2017, pp. 330–341.

[33] N. Berjab, H. H. Le, C.-M. Yu, S.-Y. Kuo, and H. Yokota, ''Abnormal-node detection based on spatio-temporal and multivariate-attribute correlation in wireless sensor networks,'' in *Proc. IEEE 16th Int. Conf Depend., Autonomic Secure Comput. (DASC)*, Aug. 2018, pp. 568–575.

[34] K. Kapitanova, E. Hoque, J. A. Stankovic, K. Whitehouse, and S. H. Son, ''Being SMART about failures: Assessing repairs in SMART homes,'' in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 51–60.

[35] A. K. Sikder, H. Aksu, and A. S. Uluagac, ''6thSense: A context-aware sensor-based attack detector for smart devices,'' in *Proc. USENIX Secur. Symp.*, 2017, pp. 397–414.

[36] P. Kumar, G. P. Gupta, and R. Tripathi, ''A distributed ensemble design based intrusion detection system using fog computing to protect the Internet of Things networks,'' *J. Ambient Intell. Hum. Comput.*, vol. 12, pp. 9555–9572, Nov. 2020.

[37] D. K. Sharma, T. Dhankhar, G. Agrawal, S. K. Singh, D. Gupta, J. Nebhen, and I. Razzak, ''Anomaly detection framework to prevent DDoS attack in fog empowered IoT networks,'' *Ad Hoc Netw.*, vol. 121, Oct. 2021, Art. no. 102603.

[38] Y. An, F. R. Yu, J. Li, J. Chen, and V. C. M. Leung, ''Edge intelligence (EI)-enabled HTTP anomaly detection framework for the Internet of Things (IoT),'' *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3554–3566, Mar. 2021.

[39] N. K. Sahu and I. Mukherjee, ''Machine learning based anomaly detection for IoT network: (Anomaly detection in IoT network),'' in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Jun. 2020, pp. 787–794.

[40] G. R. Mode, P. Calyam, and K. A. Hoque, ''Impact of false data injection attacks on deep learning enabled predictive analytics,'' in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–7.

[41] M. Wang, A. Xue, and H. Xia, ''Abnormal event detection in wireless sensor networks based on multiattribute correlation,'' *J. Electr. Comput. Eng.*, vol. 2017, Jan. 2017, Art. no. 2587948.

[42] D. Srinivasan, C. S. Chang, and A. C. Liew, ''Demand forecasting using fuzzy neural computation, with special emphasis on weekend and public holiday forecasting,'' *IEEE Trans. Power Syst.*, vol. 10, no. 4, pp. 1897–1903, Nov. 1995.

[43] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, ''Exact discovery of time series motifs,'' in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2009, pp. 473–484.

[44] S. Imani, F. Madrid, W. Ding, S. Crouter, and E. Keogh, ''Matrix profile XIII: Time series snippets: A new primitive for time series data mining,'' in *Proc. IEEE Int. Conf. Big Knowl. (ICBK)*, Nov. 2018, pp. 382–389.

[45] J. Kacprzyk, ''Group decision making with a fuzzy linguistic majority,'' *Fuzzy Sets Syst.*, vol. 18, no. 2, pp. 105–118, Mar. 1986.

[46] L. A. Zadeh, ''A computational approach to fuzzy quantifiers in natural languages,'' *Comput. Math. Appl.*, vol. 9, no. 1, pp. 149–184, 1983.

[47] *Intel Lab Data*. Accessed: Jan. 16, 2022. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

[48] M. Khayati, A. Lerner, Z. Tymchenko, and P. Cudré-Mauroux, "Mind the gap: An experimental evaluation of imputation of missing values techniques in time series," *Proc. VLDB Endowment*, vol. 13, no. 5, pp. 768–782, Jan. 2020.

NESRINE BERJAB was born in Tunis, Tunisia. She received the B.S. degree in applied network infrastructure and system administration from the National Engineering School of Carthage, Tunis, in 2012, the Dipl.-Ing. degree in software engineering from The University of Tunis El Manar, Tunis, in 2015, and the M.Sc. degree in computer science from the Tokyo Institute of Technology, Tokyo, Japan, in 2019, where she is currently pursuing the Ph.D. degree. Her current research interests include data engineering and secure and dependable IoT systems.

HIEU HANH LE received the B.S., M.E., and Dr.Eng. degrees from the Tokyo Institute of Technology in 2008, 2010, and 2015, respectively. He was a Researcher with the Yokohama Research Laboratory, Hitachi Ltd. His research interests include data engineering, information storage systems, information retrieval, and privacy. He is a member of ACM SIGMOD, IEEE CS, IPSJ, IEICE, DBSJ, and JSAI.

HARUO YOKOTA (Senior Member, IEEE) received the B.E., M.E., and Dr.Eng. degrees from the Tokyo Institute of Technology, in 1980, 1982, and 1991, respectively. He joined Fujitsu Ltd., in 1982. He was a Researcher at ICOT for the 5th Generation Computer Project, from 1982 to 1986, and Fujitsu Laboratories Ltd., from 1986 to 1992. From 1992 to 1998, he was an Associate Professor with the Japan Advanced Institute of Science and Technology (JAIST). He moved to the Tokyo Institute of Technology, in 1998, where he is currently a Full Professor. His research interests include the general research areas of data engineering, information storage systems, and dependable computing. He has been the Chair of ACM SIGMOD Japan Chapter, the Vice President of DBSJ, a Trustee Board Member of IPSJ, the Editor-in-Chief of *Journal of Information Processing*, and an Associate Editor of *The VLDB Journal*. He is currently a Board Member of DBSJ, a fellow of IEICE and IPSJ, and a member of IFIP-WG10.4, JSAI, JAMI, ACM, and ACM-SIGMOD.