Event Detection through Differential Pattern Mining in Internet of Things

Md Zakirul Alam Bhuiyan*,† and Jie Wu*
*Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
†Department of Computer and Information Sciences, Fordham University, New York, NY 10458, USA
Email: zakirulalam@gmail.com and jiewu@temple.edu

***Abstract*—Detecting an event of interest, e.g., damage in aerospace vehicles from the continuous arriving data in Internet of Things (IoT) is challenging due to the detection quality. Traditional data mining schemes are employed to reduce data that often use metrics, association rules, and binary values for frequent patterns as indicators for finding interesting knowledge. However, these may not be directly applicable to the network due to certain constraints (communication, computation, bandwidth). We discover that, the indicators may not reveal meaningful information for event detection. In this paper, we propose a comprehensive data mining framework for event detection in IoT named `DPminer`, which functions in a distributed and parallel manner (data in a partitioned database processed by one or more sensor processors) and is able to extract a pattern of sensors that may have event information with a low communication cost. To achieve this, we introduce a new sensor behavioral pattern mining technique called *differential sensor pattern* (DSP) which considers different frequencies and values (non-binary) with a set of sensors. We present an algorithm for data preparation and then use a highly-compact data tree structure (called *DP-Tree*) for generating the DSP. Evaluation results show that `DPminer` can be very useful for networked sensing with a superior performance in terms of communication cost and detection quality compared to existing data mining schemes.**

***Index Terms*—Internet of things, wireless sensor, data mining, pattern mining, event detection, resource-efficiency**

I. Introduction

With the capabilities of pervasive surveillance, Internet of Things (IoT) such as networked sensing systems have strong practical applications in many domains, e.g., structural health monitoring (SHM) for industrial machine or aerospace vehicles, chemical explosion, military surveillance, intrusion tracking [1], [2]. In these applications, high quality event detection using wireless sensing in IoT is essential. The wireless sensors in IoT produce a huge volume of dynamic data when deployed in these applications. The raw data, if accurately analyzed and transformed to usable information through data mining, can facilitate automatic and intelligent decision-making on specific events of interest (e.g., damage in aerospace vehicles), while optimizing the resource efficiency of networks. Hence, it is vital to develop methodologies to mine sensor data.

Recently, extracting knowledge from sensor data has received a great deal of attention in the data mining community [3]–[6]. Traditional data mining schemes focusing on association rules, frequent patterns, sequential patterns, clustering, and classification have been successfully used on

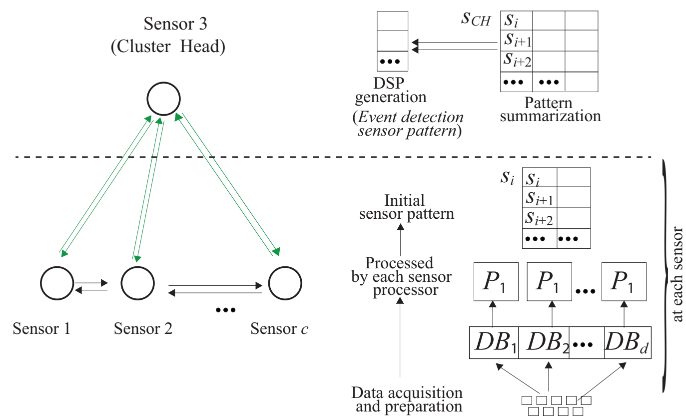


Fig. 1. The concept of `DPminer` showing interactions between sensors and their cluster head (CH) in generating a sensor pattern.

sensor data. These mining schemes are usually centralized and computationally expensive, and they focus on disk-resident transactional data. A decent number of data mining algorithms have been developed with less computational complexity [3], [7]–[9], and the process of forming patterns and producing association rules is straightforward. Metrics, rules, binary patterns, and frequent patterns are often used as *indicators* to find interesting knowledge.

Through observation, we discover that many of the indicators do not reflect meaningful information of a physical event, particularly in those applications that urgently require preprocessed data after there is an event detection indicator. We select three types of IoT data mining algorithms (associations rule based [5], associated-pattern based [7], and confidence metric-based [9]), and verify them with real datasets. We find that the communication energy cost with those algorithms is significantly high, though they often fail to detect an event. Thus, these algorithms are not directly applicable to the event detection due to resource constraints in networked systems (communication, computation, bandwidth), and the large-scale network deployment.

In this paper, we present a comprehensive sensor data mining framework for event detection in IoT named `DPminer` which functions in a distributed and parallel manner (data in a partitioned database processed by one or more sensor processors). It is able to extract a pattern of sensors that may have event information with a low communication cost. The main concept is illustrated in Fig. 1. Each partition contains a set of data acquired at the current time slot, which we call *Cases*. From *Cases*, a sensor mines different values/items and different rates of frequencies of these values, and puts in a database partition. Besides, the sensor maintains a *Controls* database/dataset that contains ranges of data (to compare with the data in *Cases*) and is defined by the event intensity in a specific application. Based on the event intensity, the sensor calculate a data pattern.

A cluster of sensors shares data patterns so that each individual sensor can calculate a sensor pattern. The sensors in a cluster coordinate with their cluster head (CH), and together, they develop a **d**ifferential data **p**attern tree structure, called *DP-Tree* in a distributed and parallel manner for data mining. The CH, along with its sensors, finds an initial *differential sensor pattern (DSP)* via the *DP-Tree*. After mining all initial DSPs, the CH provides a confirmed DSP that can ensure whether an event has occurred around some sensors or the cluster, even offering a value (e.g., $e_V > 1$) as the event indicator.

In `DPminer`, the sensors which are not in a DSP are dropped with their data from further pattern mining, thereby reducing the communication cost. Instead of finding binary frequent patterns, we find sensor patterns that come from the consideration of *different rates of frequencies and values* in the *Cases* and *Controls*. Generating such a DSP from a network can be very useful in a wide range of applications that require fine-grained monitoring of physical environments.

The major contributions of this paper are four-fold:

- We define a new type of data pattern mining for sensors in IoT, DSP, which discovers the sensors that contain event detection information. We design `DPminer` to generate the DSP for event detection.
- We propose a simplified "data preparation" algorithm, which is the first-stage data mining algorithm used to prepare the data for a tree structure in the network.
- To generate a DSP, we devise a *DP-Tree* that is developed on a sensor partitioned database in such a way that data in each sub-database can be processed by one or more sensor processors in a distributed and parallel manner.
- Finally, we validate `DPminer` in extensive simulations. We provide trade-off between sensor energy costs for communication and computation for event detection through sensor data mining in IoT. We have found that `DPminer` achieves a superior performance in terms of both communication cost and detection quality compared to existing data mining schemes.

This paper is organized as follows. Section II reviews related work. We formulate our problem in Section III. Section IV explains the `DPminer` framework. Section V presents the data preparation algorithm. Section VI develops *DP-Tree* and analysis. Evaluation through simulations is conducted in Section VII. Finally, Section VIII concludes this paper.

II. Related work

There are various data mining techniques outlined in the literature, including frequent patterns, sequential patterns, clustering, and classification. They already address numerous issues in data mining including execution time, complexity, and rule or query processing needed to mine stored (static) and/or stream data [3]–[6], [9], [10].

In the recent decades, mining association rules have been used in transactional databases. Recently, they have been applied to data mining schemes in sensor networks. Mining the associations among sensor values that co-exist temporally in large-scaled WSNs and mining spatial temporal event patterns from sensor data are proposed in [9], [11]. A behavioral pattern named Target-based Association Rules (TARs) for point-of-coverage in WSNs which aims to discover the correlation among a set of targets monitored by a WSN and uses confidence metrics is proposed [9]. In TARs, every sensor maintains an additional flash memory that increases the deployment cost.

An interesting data mining technique in wireless ad hoc networks uses a tree-based structure called Positional Lexicographic Tree (MAR-PLT for short) to mine association rules in which the event-detecting sensors are the main objects [5]. It follows a FP-growth-like pattern growth mining technique, but the two database scanning requirements and the extra MAR-PLT update operations during mining limit efficient use of this technique in handling WSN data. Association rules-based growth trees do not show satisfactory performance in WSNs in terms of communications.

A method which captures association-like co-occurrences as well as temporal correlations (linked with such co-occurrences) is used to mine *associated patterns* from sensor data streams [7]. A regular frequent pattern is proposed to find frequent sensor patterns that occur after a certain interval in the sensor database. Most of these techniques consider a binary (0/1) occurrence of the patterns in the database. Binary value (0/1) is also used for frequent pattern association. Such a binary occurrence or pattern association may fail to detect events in practice. In addition, they still require significant communication costs in terms of excessive message transmission in the WSN. Also, there is a lack of analysis of the costs in WSNs.

We observe that current data mining schemes using association rules, associated pattern, data clustering, and so on do not show satisfactory performance in terms of communication and event detection in sensors of IoT. These issues have not been specifically addressed before. Our framework `DPminer` is an attempt to overcome these shortcomings while detecting an event through a DSP.

III. Differential Sensor Pattern Mining

In this section, we describe both the network model and data mining techniques. Then, we define the problem of `DPminer`.

Let us consider a hierarchical network with a large set S of m sensors which is to be deployed for a particular monitoring application of IoT, such as SHM, $S = \{s_1, s_2, \ldots, s_m\}$. The sensors are randomly deployed in the sensing area,

TABLE I
SYMBOL DEFINITION

Symbol	Definition
D	Differential sensor patter (DSP)
F, V	data frequency, data value, respectively
rf & *mv*	rate of frequencies & median values, respectively
S_s	subset of sensors
H_h	set of tuple ($h = 1, 2 \ldots, n$)
$F_{t_k}(s_i, H_h)$	rate of frequencies in H_h of the ith sensor
$F_{t_k}(H_h)$	total frequencies in H_h
$F_{t_k}(H_h)_{rf}$	total rate of frequencies in H_h of a DB_i
$F_{t_k}(S_s, H_h)_{S_s}$	rate of frequencies in H_h of S_s
$F_{t_k}(S_s)_{rf}$	rate of frequencies in H_h of S_s in DB_i
$V_{t_k}(S_s)_{mv}$	median values in H_h of S_s in DB_i
$e_V >$	event indication based in on values
$e_F >$	event indication based in on frequencies

and they are self-organized into clusters using a clustering algorithm [12]–[14]. The underlying principle of data mining in `DPminer` involves starting from simple in-network data mining at sensors (say data preparation) to fair regional complete pattern mining at intermediate sensors (e.g., cluster heads: CHs) and finally through to global base station (BS).

We assume that the whole event detection time (Q_w) is divided into Q periods. Each period includes further q slots, i.e., $\{t_1, t_2, \ldots, t_q\}$ such that t_{k+1}-$t_k = \tau$, which is the length of each time slot. We assume that a sensor database DB can be partitioned into d sub-databases, i.e., $DB_1, DB_2, \ldots, DB_d$. One of the sub-databases (e.g., DB_1) of a sensor contains prepared data that is collected in period Q (refer to Fig. 2(i)). This arriving dataset is a large dataset which we call $Cases$. Other sub-databases are shared with the neighbors in a cluster.

Each sensor mines both different values/items (V) and different frequencies (F) of these values from $Cases$ and determines a set of tuples within time slot t_k. Here, we maintain a $Controls$ database/dataset which contains the healthy data (for comparison with the data in $Cases$). If there is an event, each tuple may have frequent values with higher event intensity information (see Fig. 2(ii)). This can be determined through comparison with data in $Controls$. In $Cases$, a set of tuples denoted by $H_h(h = 1, 2 \ldots, n)$ is defined as a subset of data (frequencies and values) of a particular sub-database. From $Cases$, we first find a rate of frequencies (*rf*) and median values (*mv*) in H_h, a subset S_s of sensors, and DB_i.

Let $F_{t_k}(s_i, H_h)$ be the rate of frequencies in H_h of the ith sensor. For example, $A = 3$ in Fig. 2(v), i.e., a value within label A has been seen three times in H_h. Toward the DSP generation, we first need to define the rate of frequencies and total values in a set of set acquired data.

Definition 1 [$F_{t_k}(H_h)$]. The rate of frequencies in a set of tuple H_h represents the total frequencies in H_h; it is given by the following equation:

$$F_{t_k}(H_h) = \sum_{s_i \in H_h} F_{t_k}(s_i, H_h) \tag{1}$$

In Fig. 4, $F_{t_k}(H_1){=}F_{t_k}(s_1, H_1){+}F_{t_k}(s_3, H_1){+}F_{t_k}(s_4, H_1){+}F_{t_k}(s_6, H_1) + F_{t_k}(s_9, H_1) = 3 + 2 + 3 + 1 + 1 = 10$.

Definition 2 [$F_{t_k}(H_h)_{rf}$]. The total **r**ate of **f**requencies that carry event information in all tuples of a DB_i is given by the following equation:

$$F_{t_k}(H_h)_{rf} = \sum_{H_h \in DB_i} F_{t_k}(H_h) \tag{2}$$

In Fig. 4, $F_{t_k}(H_1)_{rf} = 10 + 13 + 16 + 8 + 12 + 16 = 75$.

Assume that a subset S_s of sensors is working in a cluster. Then, the following equation gives the rate of frequencies in H_h:

$$F_{t_k}(S_s, H_h)_{S_s} = \sum_{s_i \in S_s} F_{t_k}(s_i, H_h) \tag{3}$$

For example, $F_{t_k}((s_2, s_3, s_5), H_2)_{S_s} = 1 + 3 + 2 = 6$ in Fig. 4. We then calculate the rate of frequencies in all of tuples of S_s in DB_i as follows:

$$F_{t_k}(S_s)_{rf} = \sum_{S_i \in DB_i} F_{t_k}(S_s, H_h). \tag{4}$$

Similarly, the total **m**edian **v**alues in all of the tuples in a subset S_s of sensors in DB_i can be calculated by the following:

$$V_{t_k}(S_s)_{mv} = \sum_{H_h \in DB_i} V_{t_k}(S_s, H_h) \tag{5}$$

In Fig. 4, $V_{t_k}(s_2, s_3, H_h)_{mv} = V_{t_k}(s_2, s_3, H_2)_{mv} + V_{t_k}(s_2, s_3, H_6)_{mv} = (3.4 + 4.6) + (6.3 + 8.3) = 22.6$.

Once we have a subset S_s of sensors' frequencies and values in DB_i, we can find the DSP by ordering the sensors based on a differential data tree structure (called *DP-Tree*). This tree is structured by each sensor, but its structural process requires the participation of both neighboring nodes and CH in a parallel and distributed manner. $V_{t_k}(S_s)_{mv}$ and $F_{t_k}(S_s)_{rf}$ are some possible criteria for developing a *DP-Tree* structure. Finally, a sensor pattern can be generalized via the *DP-Tree* using its step-by-step process.

Instead of finding a frequent sensor pattern (as is usually done in existing schemes), sensors found in DSP denoted by D come from the analysis of different frequencies and values in $Cases$ and $Controls$ of $\leftarrow S_s$. It is important to note that instead of just having a DSP, a simplified event indicator as a differentiation may also be calculated from the sensors in a DSP by the following:

$$e_V = \frac{V_{t_k}(S_s[Control])}{V_{t_k}(S_s)}, \quad e_F = \frac{F_{t_k}(S_s[Control])}{F_{t_k}(S_s)} \tag{6}$$

An event can be said to be present in the application environment if both $e_V > 1$ and $e_F > 1$. Otherwise, event information is said to be absent based on the collected data.

The differential pattern D in DB_i can be defined by the differences between $Cases$ and $Controls$ in terms of prepared data values and rates of frequencies in all sets of tuples in DB_i, i.e., *diff*$(D, DB_i) = |\{H_s(H_h, S_s)|D \subseteq S_s\}|$. A pattern D is said to be a DSP if *diff*$(D, DB_i) \geq$ *min_diff*, where *min_diff* is a user-given minimum support parameter in percentage of DB_i size in terms of size (H_s) of tuples.

Our problem is to find a D of sensors (that may report an event of interest) by mining all sets of H_h of each sensor in a cluster in a distributed manner such that a CH can finally decide whether an event has occurred in the area and report to the BS. Our objectives are to reduce the communication cost of the wireless sensor and to provide high-quality event detection.

IV. DPminer: A Distributed Data Mining Framework for Wireless Sensor in IoT

In this section, we present the `DPminer` framework. It includes a step-by-step process for finding a differential sensor pattern (DSP) in a distributed and parallel manner.

To mine data parallelly in sensors means that mining tasks are performed concurrently in multiple processing nodes, a process referred to as "auto-palatalization." However, the term "distributed" is usually associated with data mining of geographically-distributed datasets and is not concerned with computational scalability. The data can be partitioned into smaller subsets/sub-databases, and it is distributed to multiple processors [15].

Using the idea above, we consider `DPminer` as a parallel and a distributed memory-shared data mining framework for event detection in a IoT. Regarding our network model, we consider the sensor of IoT as a distributed system of m processing nodes that are collectively responsible for mining the whole prepared dataset. Each processing node is comprised of one or more processors, local memories, and limited resources, including energy. For memory sharing and processing the data on a share-basis, we also consider that a sensor database (DB) is horizontally divided into n non-overlapping partitions (where n is the number of neighboring nodes of sensor s_i). In that way, a processor of the ith sensor s_i processes almost an equal number of tuples (including different frequencies and values). Thus, we can have the DB in DB_1, DB_2,..., DB_d. We assume that each partition/sub-database is assigned to a sensor process, i.e., DB_i is assigned to sensor s_i. The example database where the dataset is partitioned into several parts can be seen in Fig. 4; each is assigned to a processor (e.g., P_1).

A CH can have some additional capacity besides its regular data mining tasks. It is responsible for performing extra sequential steps, and any of the processors can be allocated to do these tasks. This processor is called a CH/master processor P_{CH} while every other sensor's processor is called a local processor. The following key information is required to identify a DSP in a parallel and distributed environment.

Definition 3 [$F_{t_k}(S_s)_{lrf}$] The total rate of frequencies in all sets of tuples of a subset S_s of sensors that may carry event information at t_k is the sum of frequencies in all the tuples of the **l**ocal partition DB_i. It is given by the following:

$$F_{t_k}(S_s)_{lrf} = \sum_{S_s \subseteq H_h \in DB_i} F_{t_k}(H_h)_{lrf} \tag{7}$$

For example, $F_{t_k}(s_3, s_5, s_7)_{lrf}$= $F_{t_k}(H_2)_{lrf} = 13$, as shown in Fig. 4. This is because $\{s_3, s_5, s_7\}$ appears only in H_2 in the local partition P_1. Similarly, the total median value is, for example, $V_{t_k}(s_3, s_5, s_7)_{lmv}$= $F_{t_k}(H_2)_{lmv} = 40.4$.

Definition 4 [$F_{t_k}(S_s)_{grf}$] The total rate of frequencies in all sets of tuples S_s is the sum of frequencies in all the tuples of the global DB, and it is given by the following equation:

$$F_{t_k}(S_s)_{grf} = \sum_{S_s \subseteq H_h \in DB} F_{t_k}(H_h) \tag{8}$$

For example, $F_{t_k}(s_3, s_5, s_7)_{grf} = F_{t_k}(H_2) + F_{t_k}(H_{21}) + F_{t_k}(H_{22}) = 13 + 10 + 17 = 40$, as shown in Fig. 4. This is because $\{s_2, s_3, s_5, s_7, s_8\}$ appears only in H_2 in the ith local partition DB_i. Similarly, the total median value is $V_{t_k}(s_3, s_5, s_7)_{lmv} = V_{t_k}(H_2) + V_{t_k}(H_{21}) + V_{t_k}(H_{22}) = 40.4 + 39.9 + 67.3 = 147.6$.

For identifying a DSP in parallel and distributed sensors of IoT, `DPminer` executes the following steps.

Step 1. Sensor s_i carries out "data preparation" based on its acquired data and puts the prepared data into its DB_i.

Step 2. s_i scans the local database DB_i only once, and it develops an initial local *DP-Tree* structure. It maintains *DP-Tree* locally and puts the values of $F_{t_k}(S_s)_{lrf}$ and $V_{t_k}(S_s)_{lmv}$ in ith sensor s_i's **h**eader **t**able dentoed by $s_i(SH)$. It then transmits it to the CH.

Step 3. The CH sensor s_{CH} maintains a global table $s_i(GSH)$ by accumulating all values of $F_{t_k}(S_s)_{lrf}$ and $V_{t_k}(S_s)_{lmv}$ and then broadcasts the $s_i(GSH)$ table to s_i.

Step 4. s_i develops a *DP-Tree*, according to the descending order of $V_{t_k}(S_s)_{lmv}$. It then applies a compression technique to locally reconstruct *DP-Tree*.

Step 5. Sensor s_i calculates initial DSP and then sends to the s_{CH}. This DSP can be an indicator of which are the subsets of sensors that have event information.

Step 6. Finally, s_{CH} receives all the initial DSPs from all sensors and mines them, generating a final DSP.

V. Data Preparation: The 1st Stage Data Mining

In this section, we present data preparation algorithm needed towards *DP-Tree* development for event detection.

Acquired data from sensors often contains a large amount of redundancy, noise, and outliers for various reasons [3] separating out actual data from the acquired data is a rigorous task. There are various types of data extraction algorithms, cleaning methods, outlier detections, and data predictions that can help with this task using them, there is a risk of missing important data. Instead, in `DPminer` local data preparation is provided.

The idea behind the data preparation is to reduce additional data transmission and interactions by mining data, and therein, to reduce the communication cost. In the beginning of the system operation, the data preparation process diffuses the mining parameters from the BS to all nodes, and if there is a DSP, request the DSP as the event information (see Algorithm 1 for steps). These parameters include Q and t_k with slot length τ. Upon the returned message reception, the BS sets both its time slot and its time for data collection. This is highly important for synchronized data acquisition [16].

Upon receiving the mining parameters, sensor s_i executes commands, including timing and clustering. It also sets its

(i) Acquired data at t_1 — Sensor s_1

d1	0.05685	0.18652	0.12451	0.21546	0.06592	0.18652	...
d2	0.12596	0.01256	0.12981	0.26451	0.29865	0.08289	...
d3	0.01652	0.16029	0.17045	0.01421	0.02429	0.19077	...
...	...	...	...	...	...	...	...

(iii) Labeling Values

H_1	A	B	B	C	A	B	C	D	B	A	B	D	D	C	B	B
H_2	B	A	B	C	C	A	B	D	A	E	C	A	B	A	B	A
H_3	A	B	B	A	C	B	B	A	F	D	A	A	B	B	B	E
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

(ii) Data ranging

A=0.05		B=0.15		C=0.25		D=0.35		E=0.45		F=0.55		G=0.65		H=0.75		I=0.85		J=0.95	
0.00	0.09	0.10	0.19	0.21	0.29	0.31	0.39	0.41	0.49	0.51	0.59	0.61	0.69	0.71	0.79	0.81	0.89	0.91	1.00
low to high event intensity																			
low event intensity						low event intensity						high event intensity							

(iv) Calculating Frequencies

A=3	B=7	C=3	D=3		
A=6	B=5	C=3	D=1	E=1	
A=5	B=7	C=1	D=1	E=1	F=1

(v) Summarization

Total values	TA=14	TB=19	TC=7	TD=5	TE=2	TF=1
Rate of Frequencies	FA=3	FB=3	FC=3	FD=3	FE=2	FF=1

(vi) Total average values, T_{mv}= 8.5
Rate of Frequencies, F_{t_1}={3,3,3,3,2,1}

Fig. 2. The process of data preparation from refined data in sensors of IoT.

Algorithm 1: Network Interaction and Data Collection

The BS:
Broadcast (Q, Q_w, τ, clustering, DB partitioning, min_dif)
Upon receiving returned messages
for each $t_k(t_k = 1$ to $\frac{Q_w}{\tau})$
$I \leftarrow$ set sensors' identifiers within the same time slot
$U \leftarrow (t_k, I)$
Insert (U, DB)
Node s_i: (Upon receiving mining parameters)
for each ith node s_i of the IoT
Communicate with the neighbors and organize into clusters
$t_k = 1$
$t \leftarrow$ current time (t_c)
While $t_c \leq Q_w - t$
If $(t_c \leq t + (t_k \times \tau))$
Perform DB partitioning
Acquire data and buffer
Call Algorithm 2 at t_k for data preparation
else
t_k++

Algorithm 2. Data Preparation

for each node s_i
Pass the acquired data through the ProportionTest
Classify and label all the data according to the ranges (see Fig. 2(ii) and 2(iii))
Calculate frequency and value set
Make a summary of the total frequencies and values

DB_i. We assume that data mining can be performed at each time slot (i.e., close to process as the data arrives). When a set of data is acquired, data preparation starts. Algorithm 2 is presented for data preparation and its corresponding illustration is shown in Fig. 2. After data acquisition, s_i stores the set of data into its DB_i: $Cases$ dataset. A partial set of data can be seen in Fig. 2(i). Then, s_i performs a *proportion test* to refine the acquired data.

Proportion Test. As part of the data preparation algorithm, we perform a proportion test [17] (i.e., to check whether the collected data is within a given range or not). We then configure a simplified dataset $Controls$. It includes (i) a set of ranges to classify the acquired data in order to find frequencies and simplified values and (ii) a set of tuples with different frequencies and values defined/collected that can be defined by the healthy data (when there is no event in an application). The ranges are set between 0.01 and 1. Note that these ranges can be different for different applications due to the nature of sensor data, their special characteristics, as well as the intensity of an event required in a particular application.

Through the proportion test, a significant amount of unnecessary, irregular, or null data (i.e., the data having no relation to event information, not indicate the data in $Controls$) can be reduced. Although the proportion test helps to reduce irregular data, we still need redundant/duplicate data for event detection so do not skip them. The proportion test establishes a data pattern by comparing close frequencies and values between $Cases$ and $Controls$ (H_{in} : $\pi_{Case} = \pi_{Control}$ vs H_{out} : $\pi_{Case} \neq \pi_{Control}$, where H_{in} and H_{out} denote the frequencies and values that are 'in' the range and 'out' of the range, respectively). We denote the frequencies in the union of $Cases$ and $Controls$ by π. In the following equation, p_{Case}, $p_{Control}$, and p are estimates of π_{Case}, $\pi_{Control}$ and π, respectively. Then, we have,

$$z = \frac{p_{case} - p_{control}}{\sqrt{p(1-p)(\frac{1}{\pi_{case}} + \frac{1}{\pi_{control}})}}. \quad (9)$$

Under the null hypothesis of no difference in values, the square of the statistic z^2 follows the *Pearson's chi-squared test* [18]. In Fig. 2(iii), we have the label for the data for sets of tuples H_h. Based on this tuple set, every sensor calculates the total values and frequencies that are used in the *DP-Tree* development, as shown in Figs. 3(v) and 3(vi) .

After the data preparation, we can begin the second stage of the mining process which we call DSP mining process, each sensor represents itself with different data frequencies and values (instead of '0/1' values).

VI. DSP Mining Through *DP-Tree* Development

This section studies data mining in sensors of IoT through *DP-Tree* development, and generates a differential sensor pattern (DSP), and shows the event detection through the DSP.

Q_i	t_k	**Sensor Tuple** (H_h)	**Frequency rate**	**Average values**	**Total values**	**Partition**
Q_1	t_1	S_1, S_3, S_4, S_6, S_9	3,2,3,1,1	8.5,12.3,4.6,16.2,7.2	48.8	(DB_1)
	t_2	S_2, S_3, S_5, S_7, S_8	1,3,2,4,2,1	3.4,4.6,8.9,11.5,12	40.4	
	t_3	$S_1, S_2, S_5, S_6, S_8, S_{10}$	1,3,2,4,2,1,3	7.9,5.9,8.1,4.9,13.9,14.1	54.8	
	...	…	…	…		
	t_{10}	S_3, S_4, S_6, S_7	2,3,1,2	8.9,7.9,16.1,15.6	48.5	P_1
	t_{11}	$S_1, S_4, S_5, S_7, S_8, S_9$	3,2,1,1,2,3	5.6,7.9,12.3,13.6,4.6,5.6	49.6	
	t_{12}	$S_2, S_3, S_4, S_6, S_7, S_8, S_9$	2,4,2,1,1,2,1,3	6.3,8.3,13.9,8,5.9,7.1,52,12.1	113.6	
	...	…	…	…	…	
	t_{21}	S_3, S_4, S_5, S_7, S_9	4,3,2,1	9.7, 13.2, 9.8, 7.2	39.9	(DB_2)
	t_{22}	S_1, S_3, S_5, S_7, S_8	5,2,3,3,5	7.2,7.1,14.2,14.1,16.4,8.3	67.3	
	t_{23}	$S_1, S_2, S_4, S_6, S_8, S_{10}$	6,1,4,2,2,1	6.2, 12.6, 9.5, 4.5, 14.2, 11.2	58.2	P_2
	...	…	…	...	...	
Q_2	...	…	…	...	...	

Fig. 3. A sensor DB with sensor data values, sensor tuples, and each DB partition corresponding to a processor.

A. *DP-Tree Development*

We devise a data mining tree structure (called *DP-Tree*) on a partitioned database DB_i for generating a DSP. Each s_i maintains a *DP-Tree* to mine the pattern. The tree structure is composed of two segments: insertion segment and restructuring segment. Insertion segment arranges local DB_i contents into the tree, while restructuring segment restructures the tree into descending order.

1) Insertion Segment: We consider DB_i as shown in Fig. 3. We also consider that s_i can have two or more processors P_1 and P_2. In Fig. 3, the rows corresponding to P_1 mean that these rows are within DB_1. If s_i first has only one process, and the parts of its data (if any remain) may be processed by another processor of its own, or a neighboring sensor which is free of tasks at the time slot. ith DB_i is assigned to the ith respective processor, as shown in Fig. 3. s_i develops the insertion segment of *DP-Tree* in parallel.

The step-by-step development processes of *DP-Tree* (with the corresponding representation in Fig. 4) is as follows.

Step 1. The *DP-Tree* is initialized with developing ith sensor s_i's header table, denoted as $s_i(SH)^0$. Initially, it is empty (having a 'null' value), as shown in Fig. 4(i). This is because, the table for the tree is made empty after a period of event detection operation. However, the sensor pattern tuples can be kept for further analysis with additional space adjustment.

Step 2. In table $s_i(SH)^1$ as shown in Fig. 4(ii), the rows are allocated for the neighboring nodes. This is arranged according to the lexicographic order of sensor identifiers (i.e., $s_1 > s_2 >, \ldots, s_m$. Here, '>' implies the order of sensor ranks). The table is built by inserting every tuple into the DB_i one after another (see [5] for the lexicographic order). See Fig. 4(ii) for sensor ordering.

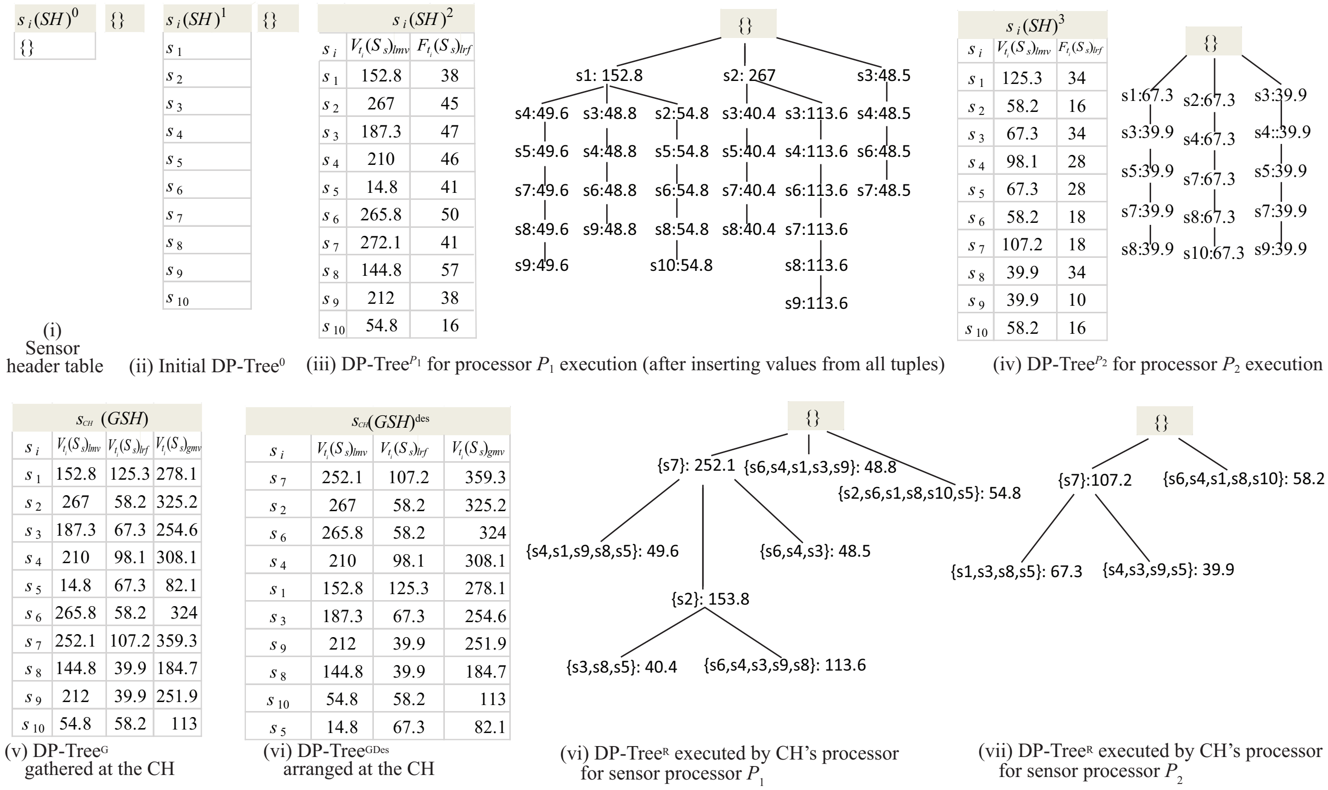
Step 3. All the tuples in each ith DB_i are inserted into the respective *DP-Trees*, following the sensor order. As shown in Fig. 4(iii) and Fig. 4(iv), the lmv values of $V_{t_k}(H_h)_{lmv}$ and the lrf values of $F_{t_k}(H_h)_{lrf}$ are calculated (refer to Fig. 2 for example prepared data) and inserted into tables $s_i(SH)^2$ and $s_i(SH)^3$, respectively. These are processed by processor P_1 and P_2, respectively. In Fig. 3, it is seen that $s_i(SH)^2$ and $s_i(SH)^3$ are complete representations of sensors DB_i, and $s_i(SH)^2$, and $s_i(SH)^3$ are constructed by lmv and lrf.

Step 4. After the insertion segment ends, the restructuring segment begins. The goal of these segments is to achieve a highly compact *DP-Tree*, which will utilize less memory. The processor P_{CH} of a corresponding CH calculates the $V_{t_k}(H_h)_{gmv}$ and $F_{t_k}(H_h)_{grf}$ values for each sensor processor which is available at each sensor's table $s_i(SH)^3$. This is a relatively small sequential step and s_{CH} performs this task. Table $s_{CH}(GSH)$ contains these values.

Step 5. When P_{CH} finishes the calculation of all $V_{t_k}(H_h)_{gmv}$ and $F_{t_k}(H_h)_{grf}$ values, it then sorts the sensors in table $s_i(GSH)$ according to the descending order of gmv values (called $s_i(GSH)^{des}$) as shown in Fig. 4(vi).

Step 6. CH sensor i_{CH} then broadcasts $s_i(GSH)^{des}$ to all of its sensors so that each sensor processor P_i facilitates restructuring as well as mining phases. s_i is enabled to *merge sort* to put the tree structure according to $V_{t_k}(H_h)_{gmv}$. For restructuring $s_i(GSH)^{des}$ to have the *DP-Tree*, a branch sorting method (BSM) is used [19]. BSM uses the merge sort to sort every path of the prefix tree. This approach first removes the unsorted paths, then sorts all the paths, and finally reinserts them into the tree. At this stage, a computationally inexpensive but effective compression process is employed. This puts the sensors with the same values of $V_{t_k}(H_h)_{mv}$ in each branch of the tree and merges them to a single node. The final *DP-Tree*, after restructuring and compression, is shown in Fig. 4(vii), after having changed from Fig. 4(i) to 4(ii) and from Fig. 4(vi) to 4(vii), respectively. Due to space limitations, we ignore further analysis of the mining process by the *DP-Tree*.

Based on the two segments of tree structure above, s_i first generates an initial DSP. If there is no event detected, DSP can be empty, i.e., $D = \{\}$. ith sensor then forwards the DSP to its CH. The CH receives all such patterns from sensors, mines the patterns, and then finally generates a DSP that may convey event information. If the system user wishes, the CH can be



$s_i(SH)^1$
s_1
s_2
s_3
s_4
s_5
s_6
s_7
s_8
s_9
s_{10}

$s_i(SH)^2$		
s_i	$V_{t_i}(S_s)_{lmv}$	$F_{t_i}(S_s)_{lrf}$
s_1	152.8	38
s_2	267	45
s_3	187.3	47
s_4	210	46
s_5	14.8	41
s_6	265.8	50
s_7	272.1	41
s_8	144.8	57
s_9	212	38
s_{10}	54.8	16

$s_i(SH)^3$		
s_i	$V_{t_i}(S_s)_{lmv}$	$F_{t_i}(S_s)_{lrf}$
s_1	125.3	34
s_2	58.2	16
s_3	67.3	34
s_4	98.1	28
s_5	67.3	28
s_6	58.2	18
s_7	107.2	18
s_8	39.9	34
s_9	39.9	10
s_{10}	58.2	16

s_{CH} (GSH)			
s_i	$V_{t_i}(S_s)_{lmv}$	$V_{t_i}(S_s)_{lrf}$	$V_{t_i}(S_s)_{gmv}$
s_1	152.8	125.3	278.1
s_2	267	58.2	325.2
s_3	187.3	67.3	254.6
s_4	210	98.1	308.1
s_5	14.8	67.3	82.1
s_6	265.8	58.2	324
s_7	252.1	107.2	359.3
s_8	144.8	39.9	184.7
s_9	212	39.9	251.9
s_{10}	54.8	58.2	113

$s_{CH}(GSH)^{des}$			
s_i	$V_{t_i}(S_s)_{lmv}$	$V_{t_i}(S_s)_{lrf}$	$V_{t_i}(S_s)_{gmv}$
s_7	252.1	107.2	359.3
s_2	267	58.2	325.2
s_6	265.8	58.2	324
s_4	210	98.1	308.1
s_1	152.8	125.3	278.1
s_3	187.3	67.3	254.6
s_9	212	39.9	251.9
s_8	144.8	39.9	184.7
s_{10}	54.8	58.2	113
s_5	14.8	67.3	82.1

Fig. 4. Distributed and parallel development of *DP-Tree* in sensors of IoT.

enabled to provide a value as an event indicator, which can be calculated by the combination e_V and e_F. If $(e_V + e_F) > 1$, an event has occurred around those sensors in the DSP. In the case of an event, the CH may request the sensors , which are in the DSP, for the data, which are in the DSP.

B. Computation and Communication Tradeoff

Initially, sensor s_i has task of data preparation. Let c_{cr}, c_{in}, $c_{F_{t_k}}$, and $c_{V_{t_k}}$ be the computation costs of data cross-checking with data in $Controls$ through the ProportionTest, refined data insertion, total frequencies, and values computations. Then, the total computation cost for data preparation is $C_{dp} = c_{cr}+c_{in}+c_{F_{t_k}}+c_{V_{t_k}}$. We have DB_i of sensor s_i. Let V and F be the total number of values and frequencies in DB_i respectively. Assume that the average computation costs to scan one tuple from DB_i and to insert it into the *DP-Tree* are c_S and c_I, respectively. Therefore, the total cost to scan all tuples from DB_i and insert them into the *DP-Tree* is $C_c = V \times F \times (c_S + c_I)$. The total cost for the CH is $C_h = c_{GSH} + c^{des}_{GSH} + c_{mine}$ for tasks in tables $s_{CH}(GSH)$ and $s_{CH}(GSH)$ and in DSP mining. $C_a = c_m + c_{BSM} + c_{P_i}$ is the extra computation cost required for merge sort c_m, BSM sort (c_{BSM}), and initial sensor pattern generation (c_{P_i}). The total computation cost for *DP-Tree* and the DSP generation is given by:

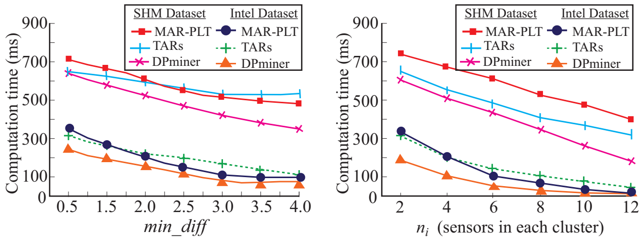
$$C_T = C_{dp} + C_c + C_h + C_a \tag{10}$$

The communication cost in the sensor of IoT. Recall that *DP-Tree* in `DP-miner` functions in a cluster. We first find cluster-wise energy costs for communication. Assume a cluster denoted by S_c contains a total of n_i sensors. Then, the total energy in a cluster, denoted by $cost(S_c)$, is given by the following:

$$cost(S_c) = X \cdot e_T + (n_i-1)X \cdot e_R + (n_i-1)\frac{n_t}{2}(e_T+e_R) \tag{11}$$

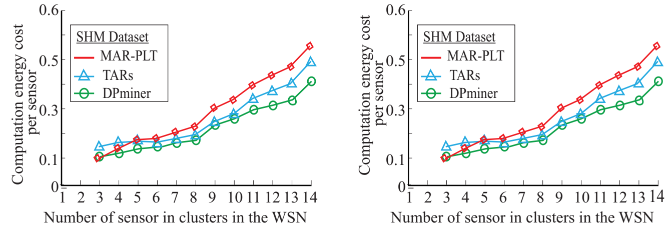
where e_T and e_R are the energy costs for transmitting and receiving X data and n_t is time length. The first two terms at the right side of (11) are , respectively, the energy costs required when a CH broadcasts its time data to its sensors or the BS, and when all the sensors receive the broadcasts, respectively. The last term is the energy consumption when the (n_i-1) sensors in the cluster transmit back their response (including connection establishment), table data transmission, sensor pattern transmission, and all data transmission if it is in the DSP; a CH may receive the request for data transmission.

From (11), we can get $cost(n_i) = cost(S_c)$, indicating that the energy consumption of a cluster is only associated with n_i sensors in a cluster. When m sensors are partitioned into equal-sized clusters of size l, then the number of clusters $isz = m/l$. The optimal cluster size [12] can be obtained by looking for l that minimizes the *average energy cost per node*, defined thusly:



(i) Computation time vs. *min_diff* (ii) Computation time vs. n_i

Fig. 5. Average computation time in different network data mining schemes for executing to the GNTVT SHM dataset and Intel dataset.



(i) Computation energy in different schemes (ii) Computation energy in different schemes

Fig. 6. Communication cost in data mining in the wireless sensor: (i) in `DPminer`; (ii) comparison between `DPminer`, MAR-PLT, and TARs.

$$Cost(s_i) = \frac{z.cost(l)}{m} + \frac{1 - l/m}{l - 1}\kappa \tag{12}$$

where κ is a constraint on the overlapping sensor nodes in the cluster [12].

VII. Performance Evaluation

A. Methodology

We evaluate the performance of `DPMiner` and its *DP-Tree* development for a DSP generation. The objective is to verify its ability in terms of communication and computation cost, and the quality of event detection. We conduct an extensive set of simulations for the mining process using *DP-Tree*.

We consider two sets of large datasets for the evaluation, and we evaluate the performance of `DPminer` in heterogeneous sensors in IoT. The first dataset containing real sensor data is from the Intel Berkeley Research Lab [20] and has been widely used [5]. This consists of tuples from 54 sensors and 84600 time slots (one month). The second dataset we used is collected by an SHM system deployed on the Guangzhou National TV tower (GNTVT) [21]–[23]. It consists of a set of 800 wired acceleration sensors data, collected in 273000 time slots. However, to see the DSP mining performance in sensors of IoT, we consider the GNTVT SHM dataset in the 200-sensor case.

Considering recent advancements of IoT, as modeled before, we consider that some sensors could have greater memory and more processors than others. Each DB is distributed among the sensors, and the processor in the node has complete access to its portion of the database. Simulations are performed with Omnet++ simulation tool within a $50m \times 500m$ rectangular field, taking into account the SHM environment, e.g., a high-rise building, bridge, aircraft, etc. The hardware constants for the processor and transceiver are from the Intel Xscale PXA271. The Imote2 uses a CC2420 radio chip for wireless communication. We model each sensor with six discrete power levels in the interval {-10dBm, 0dB}. We adopt similar configurations from an improved log-normal path loss model [24] and a synchronized data collection method [17] only for data forwarding. For the sake of convenience, we normalize the communication cost from 0% to 100%.

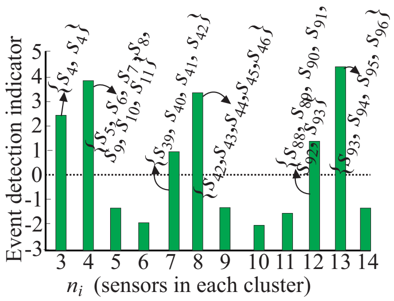
For observing the presence of an event, we consider the GNTVT SHM dataset and give different levels of event injection (damage information) at different sensor locations (by modifying the input signal randomly in the data sets of (5-10)th sensors, (41-45)th sensors, (90-95)th sensors, and (170-175)th sensors). For comparison, we consider two other sensor network data mining schemes: MAR-PLT [5] and TARs [9].

B. Performance Results

1) Computation Cost: In the first set of simulations, we observe the average computation time in generating a DSP in `DPminer`. We gather the time for two data set computations in sensor IoT. The total computation time is composed of the time for data preparation, *DP-Tree* development (including data insertion, tree restructuring, delay in data broadcasting/receiving between the CH and sensors), and finally, DSP generation. The results for the two data sets at their respective min diff parameter settings (defined in Section III) are in Fig. 5(i). We vary the *min_diff* parameter from 1.0 to 4.0. It is found that the computation time in `DPminer` is a little lower compared to that of MAR-PLT and TARs. We note that the computational load is almost equally distributed among all the processors in a cluster for the two data sets. In Fig. 5(ii), it is evident that the computation time decreases when the number of processors increases. Importantly, the rate of decreases is faster in `DPminer` than the rates found in MAR-PLT and TAR.

2) Energy Cost of the sensor in IoT for DSP Mining: Here, we discuss the results by using the SHM dataset. The energy cost of the sensor for communication in `DPminer` is shown in Fig. 6(i). Based on parameters for sensors and clusters, we demonstrate the communication energy cost for various cluster sizes, when the transmission power e_T is set from $e_T = 1e_R$ to $e_T = 6e_R$. This is because the communication cost dominates the energy cost in a wireless sensor.

With the increase of sensors in clusters, the communication cost decreases slowly at first; then, it increases speedily. Some observations are as follows: when the number of sensors (n_i) in a cluster is small to medium (e.g, 3 to 6), the sensors have low communication tasks for *DP-Tree* development; when n_i is medium to high (e.g, 6 or more), there are high communication tasks for *DP-Tree* development. The comparison of different schemes in terms of communication energy cost can be seen in Fig. 6(ii). We find that `DPminer` consumes a lower amount of energy than either MAR-PLT and TARs. Both MAR-PLT and TARs apply a lot of association rules between



Event detection via DSPs in DPminer

Fig. 7. Performance of the event detection through differential pattern mining.

sensors and interactions, and the tree development process in them requires a significant communication cost (which is not investigated in their works).

Due to space constraints, we omit the analysis of computation energy cost. In an observation, we find that both the computation and communication energy costs are steady at first and then gradually increase when the size of *DP-Tree* increases. With similar computation energy costs, `DPminer` significantly reduces the communication energy cost in data mining compared to both MAR-PLT and TARs.

3) Performance on the Event Detection: Finally, we report an interesting result about event detection performance in `DPminer` regarding the situations of event detection in AR-PLT and TARs. Recall that we have provided event information injection into some of the sensors' data. Corresponding clusters containing these sensors should have a DSP. Fig. 7 shows the performance on the event detection in different clusters in `DPminer`. Here, a detection indicator is calculated by average values of $(e_V + e_F)$ in different time slots within a given period of time in the wireless sensor of IoT. We find the indicator cluster-wise since the *DP-Tree* is developed between sensors in each cluster in a distributed and parallel manner.

VIII. Conclusion

In this paper, we have proposed `DPminer`, a comprehensive data mining framework for wireless sensoring in IoT which functions in a distributed and parallel manner and is able to extract a pattern of sensors that have event information. It is a unique mining framework which works on sensing actual values and providing important values as outputs (rather than "0/1" binary decision) for event detection. `DPminer` hints that if an application user wishes to have further analysis on the event, such outputs can be crucial. Thus, it can be useful for many IoT applications. We have validated that with a lower or similar computational time in generating a sensor pattern for event detection, `DPminer` can significantly reduce the energy for communication in IoT. Applying the differential sensor mining technique with a machine-learning approach and in big data environments will be our future work.

Acknowledgment

This research was supported in part by NSF grants CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, and ECCS 1231461.

References

[1] M. Z. A. Bhuiyan, J. Wu, G. Wang, , and J. Cao, "Sensing and decision-making in cyber-physical systems: The case of structural health monitoring," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2016, http://dx.doi.org/10.1109/TII.2016.2518642.

[2] M. Z. A. Bhuiyan, G. Wang, and A. V. Vasilakos, "Local area prediction-based mobile target tracking in wireless sensor networks," *IEEE Transaction on Computers*, vol. 64, no. 2, pp. 1968–1982, 2015.

[3] A. Mahmood, K. Shi, S. Khatoon, and M. Xiao, "Data mining techniques for wireless sensor networks: A survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2013, pp. 1–24, 2013.

[4] H. J. Woo, S. J. Shin, K. H. Joo, and W. S. Lee, "Finding context association rules over sensor-actuator data streams," *IEEE Transaction on Computers*, vol. 62, no. 7, pp. 74–77, 2014.

[5] A. Boukerche and S. Samarah, "A novel algorithm for mining association rules in wireless ad hoc sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 865–877, 2008.

[6] C. Nawapornanan and V. Boonjing, "An efficient algorithm for mining complete share-frequent itemsets using bittable and heuristics," in *Proc. of ICMLC*, 2012, pp. 96–101.

[7] M. Rashid, I. Gondal, and J. Kamruzzaman, "Mining associated patterns from wireless sensor networks," *IEEE Transaction on Computers*, vol. 64, no. 7, pp. 1998–2011, 2014.

[8] K. Romer, "Distributed mining of spatio-temporal event patterns in sensor networks," in *Proc. of DCOSS*, 2006, pp. 103–116.

[9] S. Samarah, B. Azzedine, and S. Alexander, "Target association rules: A new behavioral patterns for point of coverage wireless sensor networks," *IEEE Transaction on Computers*, vol. 60, no. 6, pp. 879–889, 2011.

[10] S. Tanbeer, C. Ahmed, and B. Jeong, "An efficient single-pass algorithm for mining association rules from wireless sensor networks," *IETE Technical Review*, vol. 26, no. 4, pp. 280–289, 2009.

[11] K. Loo, I. Tong, and B. Kao, "Online algorithms for mining interstream associations from large sensor networks," in *Proc. of PAKDD*, 2005, pp. 143–149.

[12] X. Liu, J. Cao, S. Lai, C. Yang, H. Wu, and Y. Xu, "Energy efficient clustering for WSN-based structural health monitoring," in *Proc. of IEEE INFOCOM*, 2011, pp. 1–9.

[13] M. Z. A. Bhuiyan, G. Wang, J. Wu, X. Xiaofei, and X. Liu, "Application-oriented sensornetwork architecture for dependable structural health monitoring," in *Proc. of IEEE PRDC*, 2015, pp. 134–147.

[14] M. Z. A. Bhuiyan, G. Wang, J. Cao, , and J. Wu, "Deploying wireless sensor networks with fault-tolerance for structural health monitoring," *IEEE Transaction on Computers*, vol. 64, no. 2, pp. 382–395, 2015.

[15] F. Stahl, M. Gaber, and B. Bramer, "Scaling up data mining techniques to large datasets using parallel and distributed processing," in *Business Intelligence and Performance Management*, 2013, pp. 243–259.

[16] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale," in *Proc. of ACM SenSys*, 2013, pp. 1–14.

[17] P. Armitage, G. Berry, and J. N. S. Matthews, *Statistical Methods in Medical Research*. 4th Edn: Wiley-Blackwell, 2002.

[18] 2015, https://en.wikipedia.org/wiki/Pearson's_chi-squared_test.

[19] S. Tanbeer, C. Ahmed, and B. Jeong, "Efficient single-pass frequent pattern mining using a prefix-tree," *Information Sciences*, vol. 179, no. 5, pp. 559–58, 2009.

[20] Intel lab data, http://db.csail.mit.edu/labdata/labdata.html.

[21] Http://www.cse.polyu.edu.hk/benchmark/ (last access: 2015).

[22] Y. Ni and H. Zhou, "Guangzhou new tv tower: integrated structural health monitoring and vibration control," in *Proc. of the 2010 Structures Congress of American Society of Civil Engineers*, 2010, pp. 3155–3164.

[23] Y. Ni, Y. Xia, W. Liao, and J. Ko, "Technology innovation in developing the structural health monitoring system for Guangzhou New TV Tower," *Structural Control and Health Monitoring*, vol. 16, no. 1, pp. 73–98, 2009.

[24] Y. Chen and A. Terzis, "On the implications of the log-normal path loss model: An efficient method to deploy and move sensor motes," in *Proc. of ACM SenSys*, 2011, pp. 26–39.