View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Federation ResearchOnline

Procedia Computer Science
Volume 29, 2014, Pages 124–133
ICCS 2014. 14th International Conference on Computational Science

# A Technique for Parallel Share-Frequent Sensor Pattern Mining from Wireless Sensor Networks

Md. Mamunur Rashid, Iqbal Gondal and Joarder Kamruzzaman
*Faculty of Information Technology*
*Monash University, Melbourne Australia*
*{md.rashid, iqbal.gondal, joarder.kamruzzaman}@monash.edu*

**Abstract**
WSNs generate huge amount of data in the form of streams and mining useful knowledge from these streams is a challenging task. Existing works generate sensor association rules using occurrence frequency of patterns with binary frequency (either absent or present) or support of a pattern as a criterion. However, considering the binary frequency or support of a pattern may not be a sufficient indicator for finding meaningful patterns from WSN data because it only reflects the number of epochs in the sensor data which contain that pattern. The share measure of sensorsets could discover useful knowledge about numerical values associated with sensor in a sensor database. Therefore, in this paper, we propose a new type of behavioral pattern called share-frequent sensor patterns by considering the non-binary frequency values of sensors in epochs. To discover share-frequent sensor patterns from sensor dataset, we propose a novel parallel technique. In this technique, we develop a novel tree structure, called parallel share-frequent sensor pattern tree (PShrFSP-tree) that is constructed at each local node independently, by capturing the database contents to generate the candidate patterns using a pattern growth technique with a single scan and then merges the locally generated candidate patterns at the final stage to generate global share-frequent sensor patterns. Comprehensive experimental results show that our proposed model is very efficient for mining share-frequent patterns from WSN data in terms of time and scalability.

*Keywords:* Wireless sensor networks, data mining, knowledge, share-frequent sensor pattern, parallel processing, distributed system

## 1 Introduction

A WSN consists of a large number of heterogeneous or homogeneous nodes usually called as sensor nodes which communicates through wireless media to the concentrator node called as sink node and works cooperatively to monitor the environment. Sensors are integrated as an ad-hoc fashion to originate a network that is able to deliver the detected event in a multi-hop transmission to the sink periodically or if they meet a particular predicate [2]. In general, in this mode of transmission WSNs

© The Authors. Published by Elsevier B.V.
doi:10.1016/j.procs.2014.05.012

generates a large amount of data in the form of stream. Data mining techniques have recently received a great deal of attention to extract interesting knowledge from these stream data. As a result, the stream nature of the data, the limited resources, and the distributed nature of sensor networks bring new challenges for the mining techniques that need to be addressed.

Data mining techniques have shown to be a promising tool to improve WSN performance and quality of services (QoS) [3]. Knowledge discovery in WSN has been used to extract information about the surrounding environment, that are deduce from the data reported by sensor nodes and behavioral patterns about sensor nodes, which are evolved from meta-data describing sensor's behaviors. Loo et al. [7] and Romer et al. [8] have focused on extracting pattern regarding the phenomenon monitored by the sensor nodes, where the mining techniques are applied to the sensed data received from the sensor nodes and stored in a central database. Recently proposed sensor-association rules in [6], where patterns are extracted regarding the sensor nodes rather than the area monitored by the WSN. An example of sensor association rules could be ($s_1$, $s_2$ => $s_3$, 85%, $\lambda$) which means that if sensor $s_1$ and $s_2$ detect events within time $\lambda$, then there is 85% of chance that $s_3$ detects events within same time interval. Support metric-based sensor association use occurrence frequency of pattern as criteria, but, the occurrence frequency of a pattern may not be an appropriate criterion for finding significant patterns because it only reflects the number of epochs in the database which contain that pattern. Share-frequent sensor pattern mining can find more useful and realistic knowledge from sensor database by considering the non-binary frequency values of sensors in epochs. For example, for a particular time slot sensor $s_1$ trigger 4 times, sensor $s_2$ trigger 3 times and sensor $s_3$ trigger 5 times. But the support measure value cannot analyze the exact number of trigger per time slot. Share measure can provide useful knowledge about the numerical values that are typically associated with the epoch sensors. Though mining share-frequent sensor patterns from sensor stream data is extremely important in real-time applications, no such mining technique is proposed yet. Moreover, since WSNs generate a large amount of data, therefore, when mining such kind of vast data, more efficient approaches such as parallel and distributed computing (besides serial approach) are needed.

Motivated from the above demand, it is necessary to develop analytically the share-frequent sensor patterns that will generate time share relations among the sensors in the sensor networks. Therefore, in this paper, we develop a novel parallel technique for mining share-frequent sensor patterns from WSNs that overcomes the single processor and main memory based computation. In this technique, we construct a single-pass tree structure, called the parallel share-frequent sensor pattern tree (PShrFSP-tree) that can capture important knowledge from the stream contents of sensor data of each local site in a very compact manner. Then, using FP-growth [5] like pattern-growth approach, PShrFSP-tree can efficiently mine the candidate patterns from the sensor dataset of each local site independently. Finally, the global share-frequent sensor patterns are computed from the locally generated candidate patterns. Extensive performance study shows that our proposed technique is very efficient in discovering share-frequent sensor patterns over sensor data stream.

The remainder of this paper is organized as follows. In Section 2, we describe the related works. In section 3, we discuss the problem of mining share-frequent sensor patterns in parallel and distributed environment over sensor data stream. In Section 4, we represent our proposed parallel and distributed framework. In Section 5, our experimental results are presented and analyzed. Finally, Section 6 concludes the paper.

## 2 Related Works

Association rule mining [4], to generate patterns from sensor nodes in WSN, finds correlations among the objects that occur in the same context in transactional database. Initially mining association rules was proposed for transactional database, but recently it has been applied to various domains. Loo et al. [7] studied the problem of mining the associations among sensor values that co-exist temporally in large-scaled wireless sensor networks. Romer [8] addressed the problem of mining spatial temporal event patterns from sensor data.

Several algorithms in the literature have been proposed to mine frequent patterns from transactional databases. Han et al. [5] proposed a tree structure named FP-tree (Frequent Pattern-tree) and an algorithm called FP-growth which removes the candidate generation-and-test problem of Apriori algorithm. It needs only two database scans. Boukerche et al. [6] presented Positional Lexicographic tree (PLT) to store a sensor's event detecting status. PLT follows FP-growth like pattern growth mining technique. However, the requirement of two database scans for this kind of trees (e.g., FP-tree and PLT) is not suitable for generating association rules from WSN data. On the other hand, mining PLT needs an extra mapping mechanism to transform sensor meta-data to a position vector. Since, in real scenario the data mining methods need to process large databases,, therefore, researchers focused on large-scale parallel and distributed frequent pattern mining techniques [14, 15, 16]. In frequent pattern mining problem only the binary occurrence of the patterns are considered. However, the non-binary values of the patterns can discover useful knowledge from the database.

Carter et al. [9] first presented a share-confidence framework to detect share-frequent itemsets. The ShFSM (Fast Share Measure) algorithm used level closure property instead of downward property which to improve the past algorithms [10]. But, ShFSM generates too many candidates at each pass so it is time consuming approach. The DCG (Direct Candidate Generation) algorithm [11] overcomes the ShFSM algorithms problem by generating candidate directly without the pruning and joining steps in each scan, and it generates less candidates than ShFSM. Although, DCG can conserve downward closure property, its main problem is related to number of database scans, which depends on the maximum number of candidate length and it drives extremely large additional candidate patterns. Ahmed et al. [12] proposed ShrFP-Tree (Share–frequent pattern tree) for share-frequent pattern mining approach which deletes the problems of DCG algorithms and finds more effective than candidate set generation-and-test approach ,which still needs three database scans. In [13], a novel tree structure was proposed which needs only two database scan to calculate the complete set of share-frequent mining and have "build once and mine many" property for interactive mining [17].

Existing all share-frequent pattern mining techniques [10-13] are single processor based techniques and they need multiple database scans to mine share-frequent patterns from transactional database. Since, WSNs generate huge amount of data, none of the existing algorithm can effectively mine share-frequent sensor patterns from the stream of sensor data. Therefore, here we propose a parallel and distributed framework to mine share-frequent sensor patterns from sensor dataset only with one database scan.

## 3 Share-Frequent Sensor Pattern Mining Problem in WSNs

Definition of share-frequent patterns for transactional databases, [9-11] can be enhanced to define share-frequent sensor patterns for WSNs in which the sensors themselves are the main objects.

Let $S = \{s_1, s_2, \ldots, s_p\}$ be a set of sensor in a particular wireless sensor network. We assume that the time is divided into equal-sized slots $t = \{t_1, t_2, \ldots, t_q\}$ such that $t_{j+1} - t_j = \lambda$, $j \in [1, q-1]$, where $\lambda$ is the size of the each time slot. A set $P = \{s_1, s_2, \ldots, s_n\} \subseteq S$ is called a pattern of a sensors. A sensor database, SD, is defined to be a set of epochs where each epoch is a tuple $E(E_{ts}, X)$ such that X is a pattern of the event detecting sensors that report events within the same time slot and $E_{ts}$ is the epoch's time slot. Let size(E) be the size of E i.e., the number of sensors in E. An epoch $E(E_{ts}, X)$ supports a pattern Y if $X \subseteq Y$. Frequency of the pattern Y in SD is defined to be the number of epochs in SD that support it, i.e., $Freq(Y, SD) = |\{ E(E_{ts}, X) \mid X \subseteq Y \}|$.

Definition 1: The *measure value* $MV(s_p, E_q)$, represents the trigger number of sensor $s_p$ in epoch $Eq$. For example, in Table 1, $MV(s_1, E_1) = 2$.

**Table 1:** A Sensor database (SD)

| TS | Epoch | Trigger | Total trigger | Partition |
|---|---|---|---|---|
| 1 | $s_1$ $s_2$ $s_6$ $s_7$ | 2,1,2,1 | 6 | $P_1$ |
| 2 | $s_2$ $s_3$ $s_8$ | 3,2,2 | 7 | |
| 3 | $s_1$ $s_3$ $s_5$ | 5,3,3 | 11 | |
| 4 | $s_3$ | 5 | 5 | |
| 5 | $s_2$ $s_3$ $s_4$ | 4,3,2 | 9 | $P_2$ |
| 6 | $s_1$ $s_3$ $s_5$ $s_6$ $s_7$ | 1,3,1,2,1 | 8 | |
| 7 | $s_1$ $s_4$ | 1,3 | 4 | |
| 8 | $s_1$ $s_3$ $s_5$ $s_6$ | 4,2,1,5 | 12 | |

Definition2: The *epoch measure value* of an epoch $E_q$ denoted as $eMV(E_q)$ means the total measure value of an epoch $E_q$ and it is defined by

$$eMV(\mathrm{Eq}) = \sum_{\mathrm{s_p \in Eq}} MV(\mathrm{s_p}, \mathrm{Eq}) \qquad (1)$$

For example, $eMV(\mathrm{E_2}) = MV(s_2, \mathrm{E_4}) + MV(s_3, \mathrm{E_2}) + MV(s_8, E_2) = 3 + 2 + 2 = 7$ in Table 1.

Definition 3: The *total measure value TMV (SD)* represents the total epoch measure value of all the epochs in SD. It is defined as,

$$TMV(SD) = \sum_{\mathrm{Eq \in SD}} eMV(\mathrm{Eq}) \qquad (2)$$

For example,$TMV(SD) = 62$ in Table 1.

Definition 4: The sensorset measure value of an sensorset X in epoch Eq, $sMV(X, E\mathrm{tq})$ is defined as

$$sMV(X, \mathrm{Eq}) = \sum_{\mathrm{s_p \in x}} MV(\mathrm{s_p}, \mathrm{Eq}) \qquad (3)$$

For example, $sMV(s_2\ s_3, E_2) = 3 + 2 = 5$ in Table 1.

Definition 5: The measure value of a sensorset X is defined as,

$$MV(X) = \sum_{\mathrm{Eq \in SDx}} sMV(X, \mathrm{Eq}) \qquad (4)$$

For example, $MV(s_1s_3) = sMV(s_1s_3, E_3) + sMV(s_1s_3, E_6) + sMV(s_1s_3, E_8) = 8 + 4 + 6 = 18$ in Table 1.

Definition 6: The share value of a sensorset X is defined as,

$$sh(X) = \frac{MV(X)}{TMV(X)} \qquad (5)$$

For example, $sh(\mathrm{s_1\ s_3}) = \frac{18}{62} = 0.29$ in Table 1.

Definition 7: If $minshare$ is a given minimum share threshold, then a sensorset is called share-frequent if $sh(X) \geq minshare$. Let for the SD in Table 1 *minshare* is 0.25, then $s_1s_3$ is share-frequent sensorset as $sh(\mathrm{s_1 s_3}) = 0.29$.

Definition 8: The *minimum measure* value $\min\_MV$, is defined as,

$$\min\_MV = ceiling(minshare \times TMV(SD)) \qquad (6)$$

If $minshare = 0.25$ in Table 1, then $min\ \_MV = ceiling(0.25 \times 62) = ceiling(15.5) = 16$.

So for a sensorset X, if $MV(X) \geq \min\_MV$, then we can say that X is a share-frequent sensorset.

The main challenge of facing share-frequent pattern mining area is the sensorset share does not have the downward closure property. For example, $sh(\mathrm{s_1}) = 0.2096$ in Table 1, so $s_1$ is a share-infrequent sensor in Table 1 for *minshare* = 0.25, but $sh(\mathrm{s_1\ s_3}) = 0.29$, so $s_1s_3$ is a share-frequent sensorset. Therefore, the downward closure property does not satisfy. We can maintain downward closure property by epoch-weighted measure value.

We consider a homogeneous distributed system of n nodes, denoted as $N_1, N_2, \ldots, N_n$. The sensor database SD is horizontally divided into n partitions as $SD_1, SD_2, \ldots, SD_n$. We assume that each

partition $SD_i$ is assigned to a node $N_1$. Therefore, for a pattern X, local weighed epoch-measure value and global epoch measure value is denoted as lewMV(X) and gewMV(X).
Definition 9: The local epoch-weighted measure value of a sensorset X, defined as lewMV(X), is the sum of the eMV values of all epochs containing X in the local partion P.

$$lewMV(X) = \sum_{X \subseteq \mathrm{Eq} \in \mathrm{P}} eMV(\mathrm{Eq}) \quad (7)$$

For example, lewMV($s_1s_3$) = eMV($E_2$) = 7, since $s_1s_3$ appears only in $E_2$ in the local partition $P_1$.
Definition 10: The global epoch measure value of a sensorset X, defined as gewMV(X), is the sum of the eMV values of all epochs containing X in the SD.

$$gewMV(X) = \sum_{X \subseteq \mathrm{Eq} \in \mathrm{SD}} eMV(\mathrm{Eq}) \quad (8)$$

For example, gewMV($s_1s_3$) = eMV($E_2$) + eMV($E_5$) = 7 + 9 = 16.

## 4 Proposed Technique

Assume a parallel and distributed-memory framework where each node consists of a processor, local memory and other available resources. The database is divided into n non-overlapping partitions in such a way that each node handles almost equal number of epochs. This ensures that similar amount of workload is assigned to each processor. The block diagram of our proposed technique is shown in Figure. 1. Discovering share-frequent sensor patterns in parallel and distributed environment can be performed in the following steps.

Step 1: Scan the local database only once and construct initial local PShrFSP-Tree in lexicographic order of sensors for each site. Local PShrFSP-tree maintains the local epoch-weighted measure values (lewMV) in the header table and sends it to the master node.

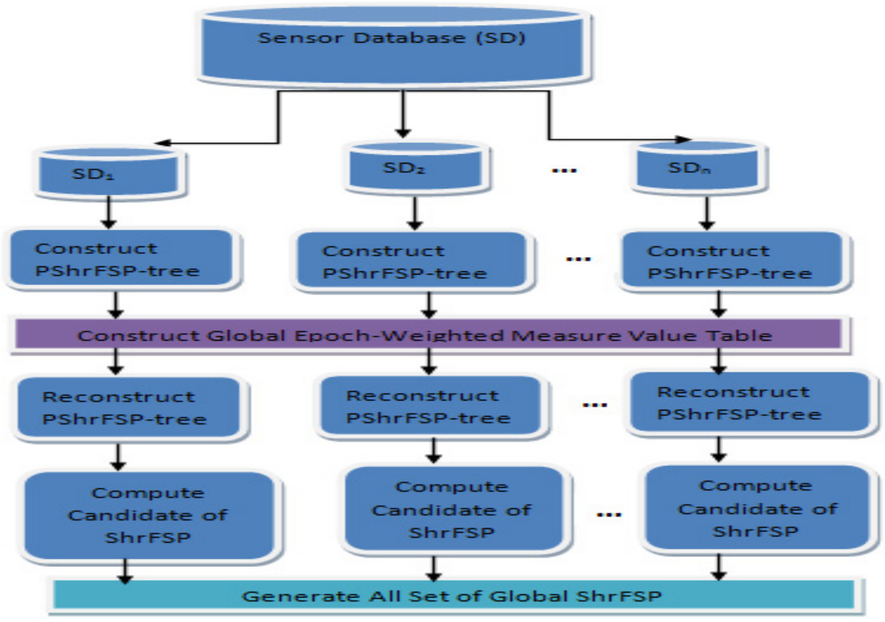
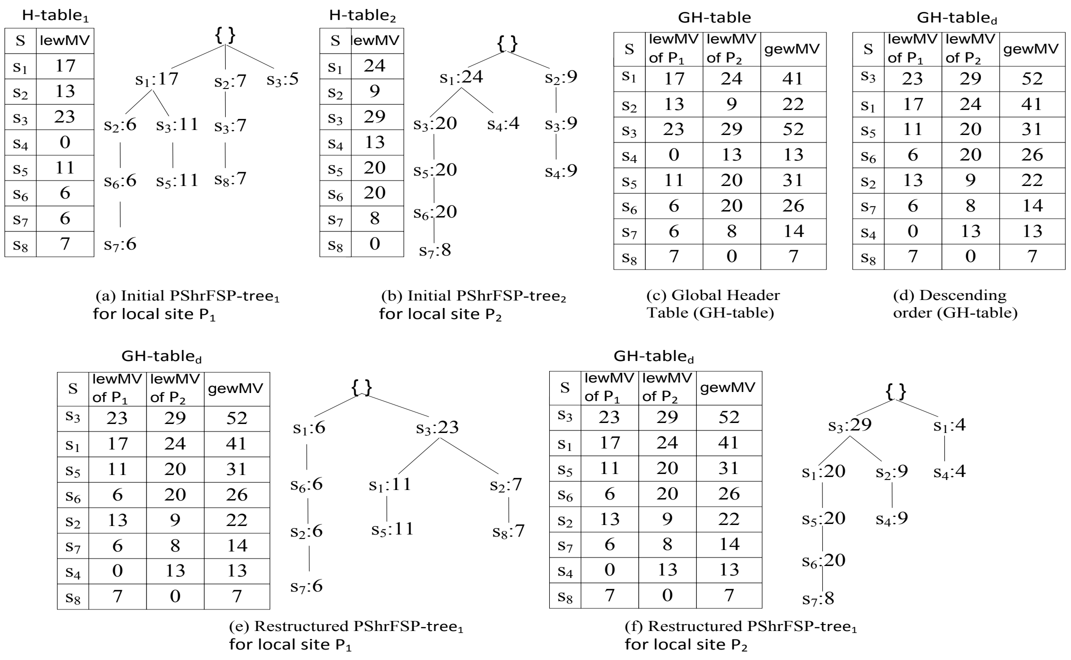


**Figure 1:** Block diagram of the proposed technique



H-table$_1$

| S | lewMV |
|---|---|
| $s_1$ | 17 |
| $s_2$ | 13 |
| $s_3$ | 23 |
| $s_4$ | 0 |
| $s_5$ | 11 |
| $s_6$ | 6 |
| $s_7$ | 6 |
| $s_8$ | 7 |

(a) Initial PShrFSP-tree$_1$ for local site P$_1$

H-table$_2$

| S | lewMV |
|---|---|
| $s_1$ | 24 |
| $s_2$ | 9 |
| $s_3$ | 29 |
| $s_4$ | 13 |
| $s_5$ | 20 |
| $s_6$ | 20 |
| $s_7$ | 8 |
| $s_8$ | 0 |

(b) Initial PShrFSP-tree$_2$ for local site P$_2$

GH-table

| S | lewMV of P$_1$ | lewMV of P$_2$ | gewMV |
|---|---|---|---|
| $s_1$ | 17 | 24 | 41 |
| $s_2$ | 13 | 9 | 22 |
| $s_3$ | 23 | 29 | 52 |
| $s_4$ | 0 | 13 | 13 |
| $s_5$ | 11 | 20 | 31 |
| $s_6$ | 6 | 20 | 26 |
| $s_7$ | 6 | 8 | 14 |
| $s_8$ | 7 | 0 | 7 |

(c) Global Header Table (GH-table)

GH-table$_d$

| S | lewMV of P$_1$ | lewMV of P$_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 23 | 29 | 52 |
| $s_1$ | 17 | 24 | 41 |
| $s_5$ | 11 | 20 | 31 |
| $s_6$ | 6 | 20 | 26 |
| $s_2$ | 13 | 9 | 22 |
| $s_7$ | 6 | 8 | 14 |
| $s_4$ | 0 | 13 | 13 |
| $s_8$ | 7 | 0 | 7 |

(d) Descending order (GH-table)

GH-table$_d$

| S | lewMV of P$_1$ | lewMV of P$_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 23 | 29 | 52 |
| $s_1$ | 17 | 24 | 41 |
| $s_5$ | 11 | 20 | 31 |
| $s_6$ | 6 | 20 | 26 |
| $s_2$ | 13 | 9 | 22 |
| $s_7$ | 6 | 8 | 14 |
| $s_4$ | 0 | 13 | 13 |
| $s_8$ | 7 | 0 | 7 |

(e) Restructured PShrFSP-tree$_1$ for local site P$_1$

GH-table$_d$

| S | lewMV of P$_1$ | lewMV of P$_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 23 | 29 | 52 |
| $s_1$ | 17 | 24 | 41 |
| $s_5$ | 11 | 20 | 31 |
| $s_6$ | 6 | 20 | 26 |
| $s_2$ | 13 | 9 | 22 |
| $s_7$ | 6 | 8 | 14 |
| $s_4$ | 0 | 13 | 13 |
| $s_8$ | 7 | 0 | 7 |

(f) Restructured PShrFSP-tree$_1$ for local site P$_2$

**Figure 2:** Parallel Construction of PShrFSP-tree

Step 2: The global epoch-weighted measure value (gewMV) table is built by master node by accumulating all the lewMV available at each local site header table and broadcasted to each local site.

Step 3: Each local PShrFSP-tree is reconstructed according to the gewMV descending order.

Step 4: The candidate of share-frequent sensor patterns are compute for each site. Then, generate the all set of global share-frequent sensor patterns.

## 4.1 The PShrFSP-tree Construction and Reconstruction

The construction process of a PShrFSP-tree consists of two phases: insertion and restructuring. The insertion phase captures the local database contents into the tree according to a lexicographic order where lewMV values of the sensors are maintain in the header table and tree nodes and when the global epoch-weighted measure value (gewMV) is available the restructuring phase restructures the tree into the gewMV descending order. Consider the sensor database shown in Table 1. Also consider that the system consists of two nodes, i.e., there are two processors P$_1$ and P$_2$ one in each node. The database is partitioned into two parts and assigned to each respective processor, as shown in Table 1. Then all the local PShrFSP-tree executes the insertion phase in parallel. During the insertion phase, all the epochs in the local database are inserted into the respective PShrFSP-tree in lexicographic order. PShrFSP-tree maintains a header table, called H-table which consists of sensor id and lewMV of each local site. To facilitate the tree traversals, adjacent links are also maintained in our tree structure like those in FP-tree [4], but are not shown in figures for simplicity. Figures 2(a) and (b) respectively show both of the local PShrFSP-trees (i.e., PShrFSP-tree$_1$ and PShrFSP-tree$_2$) and local header tables (i.e., H-table$_1$ and H-table$_2$) after the insertion phase at processors P$_1$ and P$_2$. On can see that, PShrFSP-tree$_1$ and PShrFSP-tree$_2$ are complete representations of respective local database, and H-table$_1$ and H-table$_2$ carry the lewMV for each local sensor.

To start the restructuring phase, the gewMV for each sensor is calculated by collecting all the lewMV for each sensor available at each H-table. This is a relatively small sequential step and any one of the processors can be allocated to do this task. This processor is collated master processor, Pm (i.e. node). The gewMV calculated by the Pm (GH-table) is shown in Figure 2 (c). Once the gewMV of all sensors are calculated, sensors in the GH-table are shorted according to the gewMV descending order, which is called as GH-table$_d$ is shown in Figure 2 (d). The GH-table$_d$ is then broadcast by the Pm to all the local PShrFSP-trees in order to facilitate the restructuring phase and mining phase. When the GH-table$_d$ is available to all local sites the PShrFSP-tree starts the restructuring phase. The purpose of the restructuring phase is to achieve a highly compact PShrFSP-tree which will utilize less memory and facilitate a fast mining process. In the restructuring phase, we reorganize all the local PShrFSP-tree structures according to GH-table$_d$ order. For restructuring our PShrFSP-tree, we use BSM (branch sorting method) proposed in [17]. BSM uses the merge sort to sort every path of the prefix tree. This approach, first remove the unsorted paths and then sorted the paths and reinserted to the tree. After restructuring phase the structures of the PShrFSP-trees in Figure 2 (a) and (b) are shown in Figures 2 (e) and (f) respectively.

From the construction mechanism of PShrFSP-tree, we explore the following important properties and lemmas of an PShrFSP-tree.

**Property 1:** The *ewMV* value of any node in PShrFSP-tree is greater than or equal to the sum of total *ewMV* value of its children.

**Property 2:** PShrFSP-tree for each local node can be constructed in a single database scan.

**Lemma 1**: Given a local sensor database SD$_i$ where $i \in [1,n]$ and n = number of nodes, the complete set of all sensor projections of all epochs in SD$_i$ can be derived from PShrFSP-tree$_i$.

**Proof**: Based on the PShrFSP-tree construction process, all sensor projections of each epoch in SD$_i$ are mapped to only one path in the PShrFSP-tree$_i$, and any path from the root up to a sensor maintains the complete projection for exactly z epochs, where z is the difference between the *ewMV* of the sensor itself and the *ewMV* summation for all of its children nodes. Therefore, PShrFSP-tree$_i$ maintains a complete set of all sensor projections of each epoch for sd$_i$ only once.■

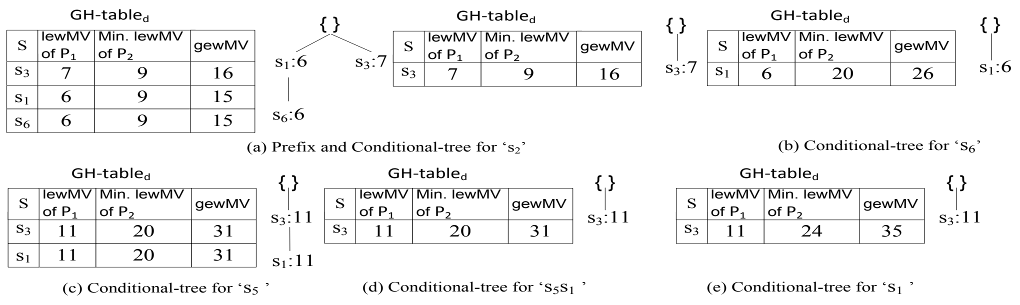
**Lemma 2:** Given a local sensor database SD$_i$ where $i \in [1,n]$ and n = number of nodes, the size of an PShrFSP-tree$_i$ (without considering the root) is bounded by $\sum_{E \in |sd_i|} |\,size(E)\,|$, where E is an epoch in SD$_i$.

**Proof:** According to property 3, an epoch E contributes at best one path in an PShrFSP-tree$_i$. It's maximum size in PShrFSP-tree$_i$ is |size(E)|. Therefore, the total size contribution of all epochs in SD$_i$ is at best $\sum_{E \in |SD_i|} |\,size(E)\,|$. Even for worst-case, where PShrFSP-tree$_i$ does not get any prefix-sharing in any node, the maximum size of PShrFSP-tree$_i$ is $\sum_{E \in |SD_i|} |\,size(E)\,|$.■

## 4.2 Mining of the PShrFSP-tree

Using property 1, we can apply a pattern growth mining in each local PShrFSP-tree to mine the candidate of share-frequent pattern from each site. Consider the mining process on PShrFSP-tree$_1$in Figure 2(e) and *minshare* = 0.25.

Since the *min_MV* for the example SD in Table 1 is 16, therefore, we start our mining process from sensor s$_1$. Considering s$_2$ as a suffix, its corresponding two prefix paths are {s$_1$s$_6$: 6} and {s$_3$ :7}, which form its prefix-tree. The prefix-tree and conditional-tree of s$_2$ is shown in Figure 3 (a). Sensors s$_1$ and s$_6$ cannot be candidate patterns with sensor s$_2$ as they have low gweMV (i.e., ewMV) with it. s$_1$ and s$_6$ ewMV value with sensor s$_2$ is 6 and minimum ewMV value must be 16 to be a candidate pattern.



GH-table$_d$

| S | lewMV of $P_1$ | Min. lewMV of $P_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 7 | 9 | 16 |
| $s_1$ | 6 | 9 | 15 |
| $s_6$ | 6 | 9 | 15 |

GH-table$_d$

| S | lewMV of $P_1$ | Min. lewMV of $P_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 7 | 9 | 16 |

(a) Prefix and Conditional-tree for '$s_2$'

GH-table$_d$

| S | lewMV of $P_1$ | Min. lewMV of $P_2$ | gewMV |
|---|---|---|---|
| $s_1$ | 6 | 20 | 26 |

(b) Conditional-tree for '$s_6$'

GH-table$_d$

| S | lewMV of $P_1$ | Min. lewMV of $P_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 11 | 20 | 31 |
| $s_1$ | 11 | 20 | 31 |

(c) Conditional-tree for '$s_5$ '

GH-table$_d$

| S | lewMV of $P_1$ | Min. lewMV of $P_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 11 | 20 | 31 |

(d) Conditional-tree for '$s_5s_1$ '

GH-table$_d$

| S | lewMV of $P_1$ | Min. lewMV of $P_2$ | gewMV |
|---|---|---|---|
| $s_3$ | 11 | 24 | 35 |

(e) Conditional-tree for '$s_1$ '

**Figure 3:** Mining Process of PShrFSP-tree$_1$

Therefore, the conditional-tree of sensor $s_2$ does not contain the sensors $s_1$ and $s_6$. So, the candidate pattern $\{s_2\ s_3\}$ and $\{s_2\}$ is generated here. The similar process is recursively occurs for other sensor which is shown in Figure 3.

In this way, each node locally generates the candidates' patterns in parallel without any inter-processor communications. Then, all local PShrFSP-tree send locally generated candidate patterns to the master node Pm. Finally, relatively small sequential step the master node accumulated the share value of each pattern and pruned if the patterns do not satisfied the given threshold. The resultant global share-frequent sensor patterns are $\{s_1\ s_6\}$, $\{s_1\ s_3\ s_5\}$, $\{s_1\ s_3\ s_6\}$, $\{s_1\ s_3\ s_5\ s_6\}$, $\{s_1\ s_3\}$ and $\{s_3\}$.

## 5 Experimental Results

In this section, we present the experimental results on mining share-frequent sensor patterns by our proposed technique. To evaluate the performance of our proposed approach, we have performed experiments on IBM synthetic dataset (T10I4D100K) and real life dataset *mushroom* and *kosarak* from frequent itemset mining dataset repository [18]. Context and objects in these datasets are similar to the epochs and sensors in the terminology of this paper. These datasets maintains binary quantity of each item for each transaction. We generated random numbers for the quantity of each item in each transaction, ranging from 1 to 10 like [10, 11]. We consider identical configuration for all nodes. Each node consists of a 2.4 GHz CPU with 4 GB memory and running on Windows 7. Communications among nodes are assured through a message passing interface. Our programs are written in Microsoft Visual C++. We assume that each database is distributed among the nodes, and that the processor in the node has complete access to its portion of the database.

At first, we examined the scalability of our proposed parallel algorithm by varying the number of processors over all above datasets. The results of the experiments are shown in Figure 4 for fixed *minshare* threshold. The y-axis of the Figure 4 shows the total execution time which includes the local PShrFSP-tree construction time, restructuring time, GH-table$_d$ construction and broadcasting time, mining time of local PShrFSP-tree and the global share-frequent sensor patterns generation time. From Figure 4, we can see that total execution time with PShrFSP-tree decreases when the number of processors increases.

Secondly, we evaluated the effectiveness of our proposed method by varying the *minshare* threshold but keeping the number of processors fixed. Figure 5 shows the result of the experiments for the above datasets. The parameter 'P' in the graph indicates the number of processors which is fixed as 4 for each datasets. It is observed from the Figure 5 that if we increase the *minshare* value, the less execution time is needed to mine share-frequent patterns.

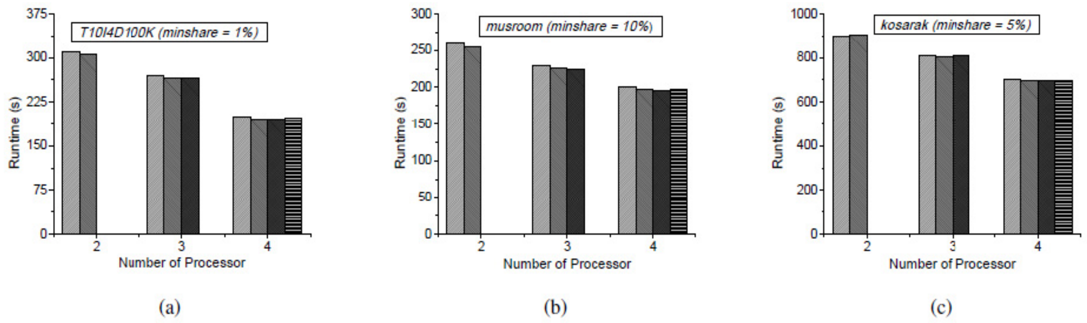


**Figure 4:** Execution time variation with number of processor on PShrFSP-tree. a) *T10I4D100K*, b) *musroom,* and c) *kosarak*

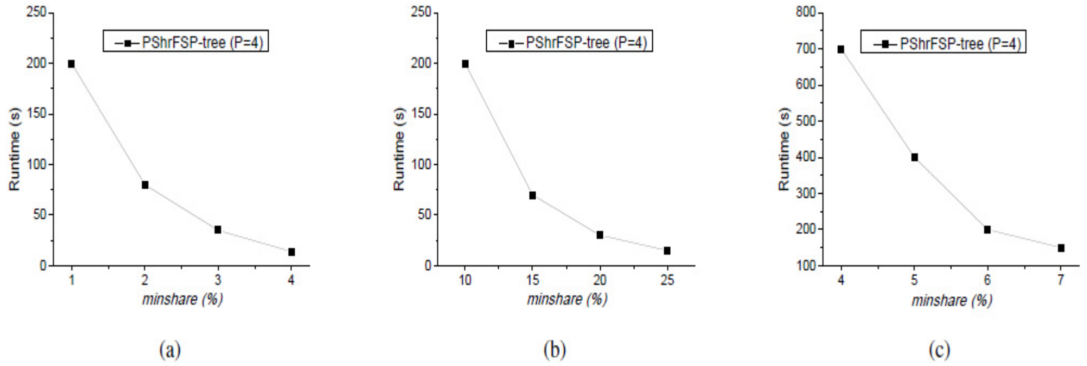


**Figure 5:** Execution time variation with *minshare* on PShrFSP-tree. a) *T10I4D100K*, b) *musroom*, and c) *kosarak*

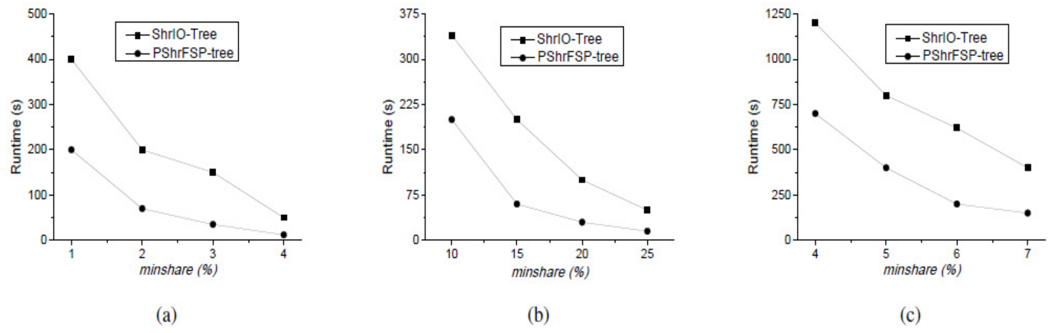


**Figure 6:** Execution time comparison: PShrFSP-tree v/s ShrIO-Tree on a) *T10I4D100K*, b) *musroom*, and c) *kosarak*

Finally, we compared the efficiency of PShrFSP-tree with that of ShrIO-Tree [13] by varying the *minshare* threshold. ShrIO-tree is a single processor based technique that was proposed to mine share-frequent pattern from transactional database. In this experiment, for PShrFSP-tree the number of processors is fixed as 4 for each dataset. The result of the experiments is shown in Figure 6, where we can see that PShrFSP-tree is significantly outperforms ShrIO with respect to execution time. The reason is that PShrFSP-tree used high degree of parallelism.

## 6 Conclusion

This paper presents a novel parallel technique for mining share-frequent sensor patterns from WSNs which overcomes the single processor based computation and is highly scalable for large sensor dataset. The PShrFSP-tree proposed in this framework significantly reduces the I/O cost by capturing the local database contents with a single scan and facilities by a fully parallelizing pattern growth mining technique with reduced inter-processor communication overhead. Extensive performance analyses show that our proposed framework is very efficient for mining share-frequent pattern from WSNs. Future research will explore ways to use the extracted knowledge to improved operational efficiency of WSN.

## References

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine, 40(8), pp. 102–114, 2002.

[2] Boukerche, A., Pazzi, R.W., Araujo, R.B.: A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications, in Proc. MSWiM, 2004.

[3] Tan , P.-N.: Knowledge discovery from sensor data. *Sensors*, vol. 23, no. 3, pp. 14–19, 2006.

[4] Agrawal, R., Imielinski, T., Swami, A.N.: Mining Association Rules between Sets of Items in large Databases. Proc. ACM SIGMOD Conference on Management of Data, pp. 207-16, 1993.

[5] Han, J., Pei, J., Yin, Y.: Mining Frequent Pattern without Candidate Generation. In Proceedings of the 2000 ACMSIGMOD international conference on Management of data, pp. 1-12, May 2000.

[6] Boukerche, A., Samarah, S.A.: Novel Algorithm for Mining Association Rules in Wireless Ad-hoc Sensor Networks. IEEE Trans, on Para. & Dis. Systems, vol.19, no. 7, pp. 865-877, 2008.

[7] Loo, K.K., Tong, I., Kao, B.: Online Algorithms for Mining Interstream Associations from Large Sensor Networks. PAKDD, pp. 143-149, 2005.

[8] Romer, K.: "Distributed Mining of Spatio-Temporal Event Patterns in Sensor Networks", EAWMS / DCOSS 2006, pp. 103-116, San Francisco,USA, June 2006.

[9] C. L. Carter, H. J. Hamilton and N. Cercone, "Share based measures for itemsets", in Proc. PKDD 1997.

[10] Y.-C. Li, J.-S. Yeh and C.-C. Chang, "A fast algorithm for mining share-frequent itemsets", in Proc. AP Web, pp. 417-428, 2005.

[11] Y.-C. Li, J.-S. Yeh and C.-C. Chang, "Direct candidates generation: a novel algorithm for discovering complete share-frequent itemsets", in Proc. FSKD, pp. 551-560. 2005.

[12] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong and Y.-K.Lee, "ShrFP-Tree: An Efficient Tree Structure for Mining Share-Frequent Patterns", AusDM '08, 2008.

[13] M.M. Rashid, M.Hossain, M.R. Karim, and B.-S. Jeong. ShrIO-Tree: A Share-Frequent Pattern Mining Approach without Candidate Generation, Proc. ICACT 2011.

[14] S.K Tanbeer, C.F Ahmed and B.S Jeong. "Parallel and Distributed Algorithms for Frequent Pattern Mining in Large Database", IETE technical review, vol 29, issue 1, 2009.

[15] Hu. Jand X. Yang-Li, "A Fast Parallel Association Rules Mining Algorithm Based on FP-Forest," *Proc. 5th Int. Symposium on Neural Networks*, pp. 40–49, 2008.

[16] A. Javed and A. Khokhar, "Frequent Pattern Mining on Message Passing Multiprocessor Systems," *Distributed and Parallel Databases*, vol. 16, pp. 321–334, 2004.

[17] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong and Y.-K.Lee, "Efficient single pass frequent pattern mining using a prefix-tree", Information Sciences, vol. 179, no. 5, pp. 559-583, 2009.

[18] Frequent itemset mining repository. Available at <http://fimi.cs.helsinki.fi/data/>.