Implementation of Weighted Rule Mining Algorithm for Wireless Sensor Networks

Kokula Krishna Hari. K, Anna University, Chennai – 600 025.

00 91 97860 31790 kk90.ssn@gmail.com

Abstract:
The Wireless Sensor Network (WSN) consists of a large number of sensor nodes that can measure and process data while communicating through wireless channels. The association rule mining techniques are used to extract hidden knowledge from the databases. Rule mining under the wireless sensor networks is a complex task because the sensor nodes produce the data in a continuous manner. The data elements are collected from various sources. The data values are integrated and passed into the rule mining process. A comprehensive framework is proposed for mining Wireless Sensor Networks, which is able to extract patterns regarding the sensors’ behaviors. The main goal of determining behavioral patterns is to use them to generate rules that will improve the WSN’s Quality of Service by participating in the resource management process or compensating for the undesired side effects of wireless communication. The proposed framework consists of 1) A formal definition of sensor behavioral patterns and sensor association rules, 2) A novel representation structure that the refer to as the Positional Lexicographic Tree (PLT) that is able to compress the data gathered for the mining process and thus allows the fast and efficient mining of sensor behavioral patterns, 3) A distributed data extraction mechanism to prepare the data required for mining sensor behavioral patterns and 4) The weighted rule mining. The rule mining is applied on a production process with temperature and pressure data. Several experimental studies have been conducted to evaluate the PLT structure and the proposed data extraction algorithms for mining wireless sensor networks.

1 INTRODUCTION

Advances in wireless technologies have led to the development of sensor nodes that are capable of sensing, processing, and transmitting. This new trend in sensor technology allows the design of Wireless Ad hoc Sensor Networks (WASNs) that consist of several sensor nodes, with the main functions of sensing the area surrounding them and sending detected events to a well-equipped node, called the sink, in multihop fashion. The detected events are transmitted to the sink periodically or based on whether or not they meet a particular predicate WASNs have proven their success in a variety of applications, especially those that require the fine-grained monitoring of physical environments that are subject to critical conditions such as fire, toxic gas leaks, and explosions. These kinds of applications introduce new challenges for WASN developers. In order to guarantee an acceptable level of quality for events’ delivery, a new class of fast, reliable, and fault-tolerant protocols for WSN needs to be developed. However, the distributed nature and the limited resources of sensor nodes, as well as the unreliability of the wireless communication, cause several delay and loss of transmitted events, which will have devastating effects on the overall quality of WASNs. Several techniques have been proposed in the literature to enhance the performance of WASNs, such as clustering, aggregation, and data fusion. In this paper, we introduce a data mining solution to extract behavioral patterns from WASNs to formulate what we call sensor association rules. The main objective of the sensor association rules is to capture the temporal relations between sensor nodes based on common intervals of activities, which mean that if we receive events from sensors $s_1$ and $s_2$, then there is a 90 percent chance of receiving an event from sensor $s_3$ within units of time. The main step in the formation of association rules is to find the patterns of sensors that co-occur together and exceed a certain frequency (these patterns are called frequent association patterns). Two major impacts of sensor association rules that benefit many applications are the ability to predict the source of future events and the ability to identify sets of temporally correlated sensors. These impacts can be used to enhance the performance of WSANs by participating in the resource management process of sensor nodes in order to cope with the sensors’ limitations and reduce the undesired effects of the wireless communication, thereby improving the Quality of Service of WSANs.

Predicting the sources of future events can also be helpful in a variety of applications such as predicting faulty nodes (for example, we are expecting to receive an event from a certain node, and it does not occur), or it may be used to identify the source of the next event in the case of the emergency preparedness class of applications. Identifying correlated sensors can also be helpful in compensating for the undesirable effects of unreliable wireless communication such as missed...
reading and in the resource management process such as deciding which nodes can be switched safely to a sleep mode without affecting the coverage of the network. There are several challenges in mining sensor association rules.

* The first challenge is to find a formal definition for sensor behavioral patterns and rules.
* The second challenge is the need to design a data extraction mechanism that is able to collect data regarding sensors’ behavior from sensor nodes while also taking into consideration the limited resources of sensor nodes, especially their energy.
* The third challenge is the need for an efficient data structure that is able to compress the data collected from sensor networks and efficiently allow mining the required patterns, which, as we know, may require exploring an exponential search space. For example, if we have a set of n sensors, then there is a potential that $2^n$ patterns between sensors will be checked against the collected data to determine the frequency of each pattern. The contribution of this paper can be summarized as follows: First, we provide a reformulation of the association rule mining problem, a well-known data mining technique that makes it applicable for sensors’ behavioral data. The second contribution is data extraction methodologies for extracting the data required for mining sensor association rules. Third, we propose an efficient data mining algorithm for generating sensors’ behavior patterns, which uses the Positional Lexicographic Tree (PLT), a new representation structure that is able to compress the sensors’ behavioral data extracted from the sensor nodes.

2 RELATED WORKS:

In this section, we will review some of the work that has been proposed for applying data mining to sensor data. In addition, we will highlight the main techniques that have been introduced for generating association rules. Looe et al. has studied the problem of mining the associations that exist between sensor values in a stream of data reported from a wireless sensor network. They proposed a data model that stores the data and presents those in a way that makes it possible to adapt the lossy counting algorithm that makes an online one-pass analysis of the data.

In this data model, sensors are assumed to take values from a finite discrete number of values, whereas a quantization method is applied for the continuous values. The time is divided into equal-sized intervals, and a snapshot from the sensor reading is taken whenever there is an update on a sensor reading. These snapshots formulate the contexts of the database. Although taking snapshots at state changes will reduce the redundancy in the data, these snapshots occur randomly; thus, each context is associated with a weight value that indicates for how many intervals this reading is valid (that is, for how long these readings will kept unchanged). The support of the pattern is defined by the total length of non-overlapping intervals in which the pattern is valid.

Mining spatial temporal event patterns is another attempt to link the problem of mining sensor data to the association rules’ mining problem that was proposed by Romer. Romer’s approach takes into consideration the distributed nature of wireless sensor networks and proposes an in-network data mining technique to discover frequent patterns of events with certain spatial and temporal properties.

In this approach, each sensor should be aware of the events that are within a certain distance from itself (this distance may be a euclidean distance or a number of hops). The sensor then collects these events and applies a mining algorithm to discover the pattern that satisfies the given parameters. The mining parameters include a minimum support S, a minimum confidence C, a maximum scope, and a maximum history.

Each node in the network collects the events from its neighbors within the maximum scope and keeps a history of their events for the duration of the maximum history. After that, each node applies a mining algorithm to discover the frequent patterns (those that have frequency exceeding the given minimum support).

The Frequent Pattern Growth (FP-Growth) proposed by Han et al. is the core algorithm of the pattern growth approach. In this method, the database is converted into a compact representation in the form of a tree, called Frequent Pattern tree (FP-tree), which is much smaller in size than the original database.

The FP-tree is constructed in such a way that all relevant information needed in the mining process is presented in the tree structure. Note that building the tree structure requires only two scans of the database. Only one pattern can be considered at a time, and a new tree, which is referred to as a conditional structure of the pattern, is constructed from the set of frequent patterns that occur with it in the same context. This process is repeated recursively until all the frequent patterns are generated. Although the FP-Growth method has proven
its efficiency in comparison to the candidate generation approach, it has been shown in that this method is not suitable for all kinds of data.

3 ASSOCIATION RULES MINING FRAMEWORK

The proposed system is designed as two sections. They are sensor node application and Sink node application. The sensor node application is designed to capture data from the environment. The data values are passed into the sink node. The sink node application is designed to perform the rule mining application. The sensor node application is loaded in multiple destinations.

3.1 System description

The development of sensor nodes is capable of sensing, processing, and transmitting. This new trend in sensor technology allows the design of Wireless Ad hoc Sensor Networks (WSNs) that consist of several sensor nodes, with the main functions of sensing the area surrounding them and sending detected events to a well equipped node, called the sink, in multihop fashion. The detected events are transmitted to the sink periodically or based on whether or not they meet a particular predicate. WSNs have proven their success in a variety of applications, especially those that require the fine-grained monitoring of physical environments that are subject to critical conditions.

This system introduced a data mining solution to extract behavioral patterns from WSNs to formulate what the call sensor association rules. The main objective of the sensor association rules is to capture the temporal relations between sensor nodes based on common intervals of activities. For example of such a rule is \( s_1 \rightarrow s_2 \rightarrow s_3 \), 90 percent, \( \lambda \) which means that if the receive events from sensors \( s_1 \) and \( s_2 \), and then there is a 90 percent chance of receiving an event from sensor \( s_3 \) within units of time. The main step in the formation of association rules is to find the patterns of sensors that co-occur together and exceed a certain frequency (these patterns are called frequent association patterns). For instance, in the example, the rule \( s_1 \) and \( s_2 \rightarrow s_3 \) is generated from the pattern \( s_1s_2s_3 \).

3.2 Sensor network architecture

The architecture is constructed for a paper production industry process. In the production system different set of sensing devices are used such as temperature monitoring devices and pressure control devices. The temperature and pressure data values are captured continuously. The production quality is affected by the temperature and pressure values. Sometimes the quality is impacted in the paper production process. The pressure and temperature devices are used as the sensor nodes.

3.3 Sink Node

The sink node is designed to carry out rules from the sensed data values. The sink node collects data from the sensor nodes. The system is designed to extract weighted rule mining for the data. The Weighted rule mining algorithm is used for the system. The weighted support and weighted confidence values are calculated and used for the rule mining process. The system reduces the data transfer traffic and data scan time in the rule mining process. The weight values are assigned with reference to the quality factors under the production process.

3.4 A Weighted Rule Mining algorithm

The proposed algorithm for mining weighted association rules is similar to the Apriori Gen Algorithm, but the detailed steps contain some significant differences. To begin with, the also generate large weighted itemsets with increasing sizes. However, since the subset of a large itemset may not be large, mining system cannot generate candidate k-itemsets simply from the large \((k - 1)\) itemsets as in Apriori Gen. To keep the k-itemsets which may generate large j-itemsets, for \( j \leq k \), in the following passes. In order to extract such k-itemsets from the database, the use the j-support bound values.

During the operation, the j-support bounds will be calculated for all the candidate k-itemsets, where \( j \) is any number between \( k \) and the maximum possible size of the large itemset. If the count of the existing k-itemset is less than all of the j-support bounds, that it cannot be the subset of any large weighted itemsets in the coming passes, and it can be pruned. The k-itemset, which may contribute to (be subsets of) future large weighted item sets, will kept in \( C_k \).

3.4.1 Sensor Association Rules Mining

The definition of the problem of mining sensor association rules is based upon the definition of association rules proposed in the domain of transactional databases.

However, not much work has been done on how it can define association rules for wireless sensor networks, in
which the sensors themselves are the main objects in the extracted rules, regardless of their values. The values are interested if a sensor detects an event and not the value of the event. Let \( S = \{s_1, s_2, \ldots, s_m\} \) be a set of sensors in a particular sensor network. The time is divided into equalized slots \( \{t_1, t_2, \ldots, t_n\} \) such that \( t_{i+1} - t_i = \lambda \) for all \( 1 < i < n \), where \( \lambda \) is the size of each time slot, and \( T_{his} = t_n - t_1 \) represents the historical period of the behavioral data defined during the data extraction process. \( P = (s_1, s_2, \ldots, s_k) \subseteq S \) as a pattern of sensors.

**Definition 1:** A sensor database DS, the behavioral data, is defined to be a set of epochs in which each epoch is a couple \( E(Ets, P) \), where \( P \) is a pattern of sensors that report events within the same time slot. \( Ets \) is the epoch’s time slot.

**Definition 2:** Let \( P_1 \) be a pattern of sensor nodes such that \( P_1 \subseteq S \). We say that an epoch \( E(Ets, P) \) supports \( P_1 \) if \( P_1 \subseteq P \).

**Definition 3:** The frequency of the pattern \( P_1 \) in DS is defined to be the number of epochs in DS that supports it:

\[
Freq(P_1, DS) = |\{E(Ets, P) | P_1 \subseteq P\}|
\]

**Definition 4:** Sensor association rules are defined in the form of \( P' \Rightarrow P'' \), where \( P' \subseteq S \), \( P'' \subseteq S \), and \( P' \cap P'' = \emptyset \).

**Definition 5:** The frequency of the rule \( (P' \Rightarrow P'') \) represents the frequency of the pattern \( (P' \cup P'') \) in DS, whereas the confidence of the rule is defined as follows:

\[
Conf(P' \Rightarrow P'') = \frac{Freq(P' \cup P'', DS)}{Freq(P'; DS)}
\]

The sensor rule is of interest to the targeted application if its frequency and confidence are greater than or equal to a given minimum support min sup and minimum confidence percentage min conf. Note that frequency and support are used interchangeably and min sup represents the minimum number of epochs that the frequency of the rules should satisfy. The given a database of epochs generated at a particular time slot size and historical period, as well as minimum support and minimum confidence, the problem of mining sensors’ association rules is to generate all the of-interest rules present in the behavioral data. Mining association rules can be decomposed into two steps

# Generating the frequent patterns (that is, those that have frequency ≥ min sup)

# Generating the rules that satisfy the min conf restriction. Sensor association rules might be straightforward and does not take a long runtime.

### 3.4.2 Data Extraction Methodologies

The mining system will present two possible methodologies for extracting the behavioral data required for mining sensor association rules from WSANs. The first methodology is a direct reporting, in which the data are transferred to the sink without any involvement from the sensor nodes in the reporting process. The second methodology considers the overall limited resources of the network while each node is trying to optimize the number of messages that it will send. In what follows, the will describe the network architecture and then describe these methodologies in detail.

![Figure 3.1 - Detected Events for historical Period of 70 Minutes](image)

**3.4.3 Direct Reporting**

In direct reporting, the extraction process starts with the application that provides the mining parameters to the sink. These parameters include the time slot size \( \lambda \), historical period \( this \), and minimum support \( \text{min sup} \). The sink then broadcasts the time slot size and the historical period to the nodes in the network. Each node keeps track of the time. At the end of each time slot, it checks whether there is any detected event. If there is, it sends a notification message to the sink that contains its identifier and the time slot number where the event occurred (that is, an integer number is used to refer to the current time slot). At the sink node, the time is monitored. At the end of each time slot (with additional delay), the sink checks the received messages and creates an epoch for the current time slot consisting of the time slot number and a pattern of sensors’ identifiers extracted from the received messages that carry the same time slot number. Then, the sink stores this epoch in the database.

Let us consider the following: Let \( S = \{s_1, s_2, \ldots, s_6\} \) be the sensors in a particular sensor network. Let the time slot size be equal to 10 minutes and the historical period equal to 70 minutes. Assume that the extraction process is initiated at time 13:00. Fig. 2 shows the detected events within the sensor network. At the end of the first epoch (13:10), sensors \( \{s_1, s_2, \ldots, s_4\} \) send the messages, respectively, \( M_1 (1, s_1), M_2 (1, s_2), M_3 (1, s_3) \), which
contain the epoch number in which the event was detected and the sensor’s identifier. At time \(13:10 + \alpha\), the sink formulates the first epoch \(E(1, (S_1, S_2, S_3))\) and stores it in the database. The same process is repeated periodically at the end of each time slot until the end of the historical period.

**Algorithm 1: Direct reporting (Sink)**

Broadcast parameters \((T_{his}, l)\);
Slot Number = 1;
Time = current time;
While (current time <= Time + \(T_{his}\))
If (current time <= Time + (Slot Number *\(l\)))
Do nothing
Else
\(P = \{\text{sensors’ IDs of the received messages}\}\);
\(E = (\text{Slot Number}, P)\);
Insert \((E, DS)\);
Slot Number ++;
End {if}
End {while}

**Algorithm 1: Direct reporting (Node)**

Upon receiving mining parameters;
Slot Number = 1;
Time = current time;
Reported = False;
While (current time <= Time + \(T_{his}\))
If (current time <= Time + (Slot Number *\(l\)))
If ((there is a detected event) and (not Reported))
Set buffer \([\text{slot number}]\);
End [if]
Else
Slot Number ++;
End {if}
End {while}
If (number of set bits \(\geq \text{min sup}\))
List = list of time slots that have set bits;
\(M = (\text{Sensor ID, List})\);
Send M to the Sink;

**Table 3.1 Direct reporting & Distributed Extraction**

<table>
<thead>
<tr>
<th>Ts</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(S_1, S_2, S_3)</td>
</tr>
<tr>
<td>2</td>
<td>(S_1, S_2)</td>
</tr>
<tr>
<td>3</td>
<td>(S_1, S_2, S_3, S_4)</td>
</tr>
<tr>
<td>4</td>
<td>(S_1, S_2, S_3)</td>
</tr>
<tr>
<td>5</td>
<td>(S_2, S_3)</td>
</tr>
<tr>
<td>6</td>
<td>(S_1, S_3)</td>
</tr>
<tr>
<td>7</td>
<td>(S_2)</td>
</tr>
</tbody>
</table>

Table 3.1 shows the extracted epochs after a historical period of 70 minutes. Algorithm 1 shows a formal description of direct reporting.

**Algorithm 2: Distributed extraction (Sink)**

Broadcast parameters \((T_{his}, l, \text{min sup})\);
Upon receiving all messages;
For Slot Number = 1 to \((T_{his}/l)\);
\(P = \text{the set of the sensors’ identifiers within}\);
The same time slot;
\(E = (\text{Slot Number}, P)\);
Insert \((E, DS)\);
End {for}

**Algorithm 2: Distributed extraction (Node)**

Upon receiving mining parameters;
Slot Number = 1;
Time = current time;
Reported = False;
While (current time \(\leq\) Time + \(T_{his}\))
If (current time \(\leq\) Time + (Slot Number *\(l\)))
If ((there is a detected event) and (not Reported))
Set buffer \([\text{slot number}]\);
End [if]
Else
Slot Number ++;
End [if]
End {while}
If (number of set bits \(\geq\) \text{min sup})
List = list of time slots that have set bits;
\(M = (\text{Sensor ID, List})\);
Send M to the Sink;
3.4.5 PLT Tree Structure Model

The PLT and then present the PLT’s construction and mining processes. Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of sensor identifiers in a particular wireless sensor network. In this case, a lexicographic order is assumed among sensor identifiers. Recall that a lexicographic prefix tree is a tree representation in which the root node is labeled null and all other nodes represent elements in the set $S$. Each node in the tree is linked to the nodes holding the sensor identifiers that occur after the node’s identifier in the lexicographic order. The PLT of the set $S$ is constructed from the lexicographic tree of $S$ by replacing the nodes’ labels with their position values. The first sensor node $s_3$ from the left in level two has a position value equal to 2. The Rank function is used to calculate the positions of the nodes as follows: $\text{Pos}(j) = \text{Rank}(j) - \text{Rank}(i)$, where $j$ is a child node of node $i$. $\text{Rank}(\text{null})$ and $\text{pos}(\text{null})$ are set to be equal to zero. An illustrative example is provided in Fig. 3.2, which shows the PLT structure of the set $\{s_1, s_2, s_3, s_4\}$.

![PLT Tree Structure Model Diagram]

**Definition 6:** Given a set $S$ of sensors, $\text{Path}(S)$ is defined to be the list of all possible paths from the root node to any other sensor node in $S$’s PLT.

**Algorithm 3: PLT construction**

- PLT\_Construction $(DS, \text{Min\_sup})$;
- $Fs = \{\text{Frequent sensors in } DS\}$;
- Sort $Fs$ in descending order of its sensors’ frequencies;
- Assign ranks to $Fs$ elements;
- For each $E \in DS$;
- Let $E' = [s_1, \ldots, s_k]$ the frequent sensors in $E$ sorted in descending;
- Order of the sensors’ frequencies;
- $V(E') = \{\text{pos}(s_1), \ldots, \text{pos}(s_k)\}$;
- $\text{VE.Com} = \sum_{i=1}^{j-1} \text{pos}(s_j)$;
- $\text{Index} = \sum_{i=1}^{j} \text{pos}(s_j)$;
- Insert $V(E')$ to PLT [Index];
- End {for}

3.4.6 Algorithm for Mining Weighted Association Rules

The framework of the proposed algorithm for mining weighted association rules is similar to the Apriori Gen Algorithm, but the detailed steps contain some significant differences. To begin with, the also generate large weighted itemsets with increasing sizes. However, since the subset of a large itemset may not be large, the mining system cannot generate candidate $k$-itemsets simply from the large $(k-1)$-itemsets as in Apriori Gen. To keep the $k$-itemsets which may generate large $j$-itemsets, for $j \leq k$, in the following passes. In order to extract such $k$-itemsets from the database, the use the $j$-support bound values. During the operation, the $j$-support bounds will be calculated for all the candidate $k$-itemsets, where $j$ is any number between $k$ and the maximum possible size of the large itemset. If the count of the existing $k$-itemset is less than all of the $j$-support bounds, that it cannot be the subset of any large weighted itemsets in the coming passes, and it can be pruned.

The $k$-itemset, which may contribute to (be subsets of) future large weighted itemsets, will be kept in $C_k$. For example From Tables 3.2 and 3.3, the will show how the large weighted itemsets are generated from the transaction database. Suppose the wminsup (weighted minimum support threshold) is 1. To begin with, the counts of the items (1-itemsets) can be found by scanning the database once, and the maximum possible size of the large weighted itemsets can be found.

**Table 3.2 Example Database**

<table>
<thead>
<tr>
<th>Bar Code</th>
<th>Item</th>
<th>Profit</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Apple</td>
<td>100</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>Orange</td>
<td>300</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>Milk</td>
<td>400</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>Jelly</td>
<td>800</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>Mouse</td>
<td>900</td>
<td>0.9</td>
</tr>
</tbody>
</table>

**Table 3.3 Transactional Database**

<table>
<thead>
<tr>
<th>TID</th>
<th>Bar Code</th>
<th>TID</th>
<th>Bar Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 4 5</td>
<td>2</td>
<td>1 4 5</td>
</tr>
<tr>
<td>3</td>
<td>2 4 5</td>
<td>4</td>
<td>1 2 4 5</td>
</tr>
<tr>
<td>5</td>
<td>1 3 5</td>
<td>6</td>
<td>2 4 5</td>
</tr>
<tr>
<td>7</td>
<td>2 3 4 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From the example, we know that the maximum possible size of a large weighted itemsets is 4 and the counts of the items {1,2,3,4,5} are {4,5,1,6,7} respectively. Let us denote the 1-itemset containing item 1 by \(I_\alpha\). In order to find the possible large weighted itemsets, called candidate itemsets, the must keep the potentially large weighted itemsets in the current stage. In order to do this, calculate the k-support bounds.

<table>
<thead>
<tr>
<th>(X)</th>
<th>(B(X,2))</th>
<th>(B(X,3))</th>
<th>(B(X,4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>{2}</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.4 K Support Bounds for all items

The k-support bounds of the k-itemsets containing \(I_\alpha\) are given by \(B(\{I_\alpha\}; k)\): assuming that the itemsets are ordered by their weights. Similarly, the k-support bounds for the remaining itemsets are listed in Table 3.4. According to the k-support bound definition; the count of the item 1 is 4, which implies that it may be the subset of a large weighted 3 or 4-itemsets. The item 1 in the set of the candidate itemsets. By similar iteration, items 2, 4 and 5 will be kept in the set of the candidate itemsets, called \(C_1\). \(C_1\) is the set of the candidate itemsets which stores the weighted 1-itemsets with potential of being subsets of large weighted itemsets. Item 3 will be pruned because none of the k-support bounds for item 3 is less than or equal to the count of item 3. Therefore, \(C_1\) will become \{\{1\}, \{2\}, \{4\}, and \{5\}\}. By similar method, all the candidate and large weighted itemsets will be generated by the iterative method.

For the finding of large weighted 2-itemsets, the will base on \(C_1\), set of the candidate 1-itemsets to join the 2-itemsets, like as shown below: \{1, 2\},\{1, 4\},\{1, 5\},\{2, 4\}, \{2, 5\},\{4, 5\} From the above joined itemsets, the will prune those itemsets with less than the k-support bounds, and the subset of the itemsets cannot be found in the \(C_1\). After that, it will update the counts for the itemsets, and find the \(C_2\) and \(L_2\), set of the large the weighted 2-itemsets.

This process will continue until no more candidate itemsets will be found. The rules will be generated from the exhaustive search of the large itemsets, as in an algorithm for mining the weighted association rules has the following inputs and output. The main method uses the Apriori Gen with the k-support bounds. Algorithm for mining the weighted Association Rules (unnormalized binary) \((\text{MINWAL}(O))\).

**Algorithm 4.1 MINWAL \((O)\)**

Main Algorithm \((\text{wminsup}, \text{minconf}, D, w)\)

\(L=\emptyset;\)
for \(i=1; i \leq \text{size}; i++\)
\(C_i=L_i=\emptyset;\)
for each transaction do
\((SC, C_1, \text{size})=\text{Counting}(D,w);\)
\(k=1;\)
While \(|C_k| \geq k\)
\(k++;\)
\(C_k=\text{Join}(C_{k-1});\)
\(C_k=\text{Prune}(C_k);\)
\((C_k, L_k)=\text{Checking}(C_k,D);\)
\(L= L \cup L_k;\)
Rules \((SC, L);\)
Ends;

**Inputs:** A database \(D\) with the transactions \(T\) two threshold values \(\text{wminsup}\) and \(\text{minconf}\), the weights of the items \(w_i\), with ascending order, total number of transactions and the total number of the items.

**Output:** A list of rules.

4 EXPERIMENTAL ANALYSES

The measurements used to evaluate the performance of the proposed distributed extraction scheme there the number of messages needed to extract the data from the sensor network and the amount of data that there routed to the sink node. In this simulator, they have abstracted the underlying communication protocols and assumed a reliable delivery of the messages, which can be done easily by acknowledging the number of set bits in each sensor’s buffer. In addition, there are no assumptions about the nodes’ deployment. The simulator’s parameters consist of the number of nodes, the historical period, and the slot size. In addition, it has been assumed that event generation is uniformly distributed over the possible number of slots within the given historical period. In addition to the data generated by the simulator, real data extracted from a sensor network is also used in comparison studies.

This data consisted of tuples from environmental sensors of a network consisting of 54 sensors that report readings for every 30 seconds of interval. The assumed that a missed reading from a sensor is an undetected event (that is, each sensor generates an event every 30 seconds and if the event is successfully delivered to the
sink, the bit entry corresponding to this epoch is set). This can be helpful in generating patterns regarding lost readings. The measurements from the real data have been extracted using filtering routines implemented in Java. Summarizes the characteristics of the real data and the data generated using the simulator.

5 RESULT ANALYSES

The number of messages needed to report the sensors’ behavioral data and the data size accumulated at the sink for different support values. Support values have been expressed as percentages of the number of epochs present in the databases for the simulator and the real data. The number of messages and the data sizes obtained using direct reporting are given at zero support value. All the reported results show a reduction in both the number of messages and the data size while increasing the support values.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Broadcast</th>
<th>PLT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>84</td>
<td>71</td>
</tr>
<tr>
<td>4</td>
<td>87</td>
<td>73</td>
</tr>
<tr>
<td>8</td>
<td>92</td>
<td>77</td>
</tr>
<tr>
<td>10</td>
<td>96</td>
<td>81</td>
</tr>
</tbody>
</table>

Table 5.1 Traffic rate analysis

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Rule Mining</th>
<th>Weighted RM</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>800</td>
<td>37</td>
<td>46</td>
</tr>
<tr>
<td>1200</td>
<td>52</td>
<td>89</td>
</tr>
<tr>
<td>1600</td>
<td>69</td>
<td>112</td>
</tr>
<tr>
<td>2000</td>
<td>88</td>
<td>137</td>
</tr>
</tbody>
</table>

Table 5.2 Comparison of rule extraction process

The partitioning mechanism used in PLT makes it easy to locate the conditional vectors of a particular pattern instead of following the nodes’ link as in the FP-tree. PLT partitions are independent. The do not need the entire structure to be in the main memory, as opposed to the FP-tree that requires the whole structure to be in the main memory. The comparison values of the position vectors that are used to locate and insert vectors accelerate the process of accessing the PLT structure, whereas the FP-tree does not include such values. PLT requires smaller variable sizes for storing the positional values, since it uses the lexicographic distance to represent sensors, as opposed to the FP-tree that uses the actual sensors’ identifiers.

6 CONCLUSIONS AND FUTURE WORK

The proposed framework consists of a reformulation of the sensors’ association rules, distributed extraction methodology, and a compressed representation structure for sensors’ behavioral data. The new formulation captures the temporal relations between sensors. The nodes are constructed with minimum processing capacity and low transmission facility. The data processing is done at the sync node in the wireless sensor networks. The data transmission process is the complex task for the sensor nodes. The data values are transferred through the intermediate nodes. The data values are broadcasted to all nodes. The broadcast based mechanism produces high network traffic. The positioning lexicographic trees are used to reduce the data transfer process traffic. The data capturing and data transmission operations are tested with a set of sensor node applications. The data values are processed in the sync node. The system is also tested with rule mining tasks. The transaction count and interested rules for rule mining and weighted rule mining are tested. The weighted rule mining process produces more number of rules for the same support and confidence values. The system also reduces the data traffic for the PLT based transactions.

In the future the system can be adopted to perform clustering and classification on distributed environment. The clustering process can be applied on the nodes and its data sets. The system can be implemented and tested under the real paper production plant or petroleum refineries environment.

REFERENCES