2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Novel Algorithm for Mining Behavioral Patterns from Wireless Sensor Networks

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

***Abstract*—Due to recent advances in wireless sensor networks (WSNs) and their ability to generate huge amount of data in the form of streams, knowledge discovery techniques have received a great deal of attention to extract useful knowledge regarding the underlying network. Traditionally sensor association rules measure occurrence frequency of patterns. However, these rules often generate a huge number of rules, most of which are non-informative or fail to reflect the true correlation among data objects. In this paper, we propose a new type of sensor behavioral pattern called associated sensor patterns that captures association-like co-occurrences and the strong temporal correlations implied by such co-occurrences in the sensor data. We also propose a novel tree structure called as associated sensor pattern tree (ASPT) and a mining algorithm, associated sensor pattern (ASP) which facilitates frequent pattern (FP) growth-based technique to generate all associated sensor patterns from WSN data with only one scan over the sensor database. Extensive performance study shows that our algorithm is very efficient in finding associated sensor patterns than the existing significant algorithms.**

Keywords-wireless sensor networks; data mining; behavioral patterns; knowledge; stream data

I. INTRODUCTION

Wireless sensor networks (WSNs) is emerging as a promising research area which used in area monitoring, environment monitoring, industrial and machine health monitoring, waste water monitoring and military surveillance [1, 2, 3]. A WSN consists of a large number of heterogeneous or homogeneous sensor nodes that are formed to sense the environment around them and send the detected events to a well-equipped node refer to as the *sink*, in multihop fashion. The detected events are transmitted to the *sink* periodically or based upon satisfying a particular predicate or as an answer to the query [4]. In this transmission mode WSNs generates a huge of data in the form of stream. As a result, the real time data stream, limited resources and the distributed nature of sensor networks bring new challenges for data mining techniques.

Data mining techniques have recently received a great deal of attention to extract interesting knowledge from WSN [5, 6]. These techniques have shown to be a promising tool to improve WSN performance and quality of services (QoS) [7]. Loo et al. [19] and Romer et al. [21] have focused on extracting pattern regarding the phenomenon monitored by the sensor nodes, in which the mining techniques are applied to the sensed data received from the sensor nodes and stored in a central database. Sensor-association rules was proposed in [16, 17, 18] where patterns are extract regarding the sensor nodes rather than the area monitored by the WSN. An example of sensor association rules could be $(s_1, s_2 \rightarrow s_3, 85\%, \lambda)$ which means that if sensor s_1 and s_2 detect events within λ time interval, then there is 85% of chance that s_3 detects events within same time interval. However, association rule mining with real datasets is not so simple. This scheme is dependent on a constraint termed *minimum support threshold* which is use to specify minimum lower bound for the support of resulting association rules. If the *minimum support threshold* is high, then we can get high value 'knowledge'. On the other hand, when the *minimum support threshold* is low, an extremely large number of association rules will be generated, most of them are non-informative. The valid correlation among data objects are buried deep among a large pile of useless rules.

Here, we propose a new type of sensor behavioral pattern called associated sensor patterns that can be used for predicting the source of future events. By knowing the source of future event, we can detect the faulty nodes easily from the network. For example, we are expecting to get an event from a particular node, and it does not occur. It also may be used to identify the source of the next event when the behavioral pattern reveals a chain of related events. Associated sensor patterns also can identify a set of temporally correlated sensors. This knowledge can be helpful to overcome the undesirable effects (e.g., missed reading) of the unreliable wireless communications and also useful in resource management process by deciding which nodes can be switched to a sleep mode without affecting the coverage of the network. However, even though mining associated sensor patterns from WSN is extremely required in real-time applications, no such algorithm has been proposed yet.

Motivated by the above discussion, in this paper, we address the problem of finding associated sensor patterns in a sensor database. For associated sensor patterns mining, we devised a prefix-tree structure, called an associated sensor pattern tree (ASPT), which captures patterns with one scan of the sensor data streams in a highly compact manner. The main concept behind ASPT construction is to first build a prefix-tree based on the order of sensors' appearance in the database, then restructure the tree in a frequency-descending order, and finally compress the tree by merging the *same support sensor nodes* in a single node in each branch of the

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

tree. After that, we use a pattern growth approach called as ASP to mine the associated sensor patterns from ASPT. Performance study shows that the proposed approach is efficient in finding associated sensor patterns.

In summary, the main contributions of this paper are (1) We define a new type of behavioral pattern for WSNs. We refer to the new proposed pattern by associated sensor patterns, (2) We devise a novel, highly compact tree structure called ASPT that is efficient for discovering associated sensor patterns from sensor database with a single database scan, (3) We develop a noble mining algorithm ASP based on the above tree structure which can be devoted for finding associated sensor patterns over sensor data stream, and (4) We show the performance study of our proposed algorithm through extensive experimental analyses.

This paper is organized as follows. Section 2, we describe related works and in Section 3, we discuss the problem of mining associated sensor patterns in WSNs. In Section 4, we develop our proposed ASPT structure and algorithm. In Section 5, our experimental results are presented and analyzed. Finally, Section 6 concludes the paper.

II. Related Works

Association rules [8, 9] is the first data mining technique that has been used in WSNs to generate patterns related to the sensor nodes and their underlying domain. Loo et al. [19] studied the problem of mining the associations between sensor values that co-exist temporally from large-scaled wireless sensor networks. They used a data model to store the data and assumed that a sensor only takes on a finite number of discrete states where a quantization model was applied for the continuous values. The time was divided into equal-sized intervals. A snapshot from the sensor reading was taken whenever there was an update on a sensor reading and stored it in a database in the form of contexts. Then, they showed how the Lossy Counting framework [20] can be utilized to formulate a online one-pass algorithms for mining large sensor streams under the two data representations (weighted transactions and interval list). Romer [21] addressed the problem of mining spatial temporal event patterns form sensor data which was another attempt to link the association rule mining problem. In this method the distributed nature of wireless sensor networks was considered and an in-network data mining techniques were proposed to discover frequent event patterns and their spatio-temporal relationships within a sensor network, such that compact patterns rather than raw data streams would have to be transmitted from nodes to the sink.

Boukerche et al. [16, 17] introduced sensor-association rules as an attempt to extract a pattern regarding the sensor nodes, rather than the area monitored by the WSN. The main difference between sensor association rules and the other techniques was that the data used in the mining process were behavioral data (i.e., meta-data describing the nodes' activities). They also presented positional lexicographic tree (PLT) to store the sensor's event detection status. PLT tree follows a pattern growth mining technique similar to FP-growth approach [10]. The mining starts with the sensor having maximum rank by generating the frequent patterns from its PLT in a recursive way. However, construction of such kind of tree (e.g., FP-tree and PLT-tree) requires two database scans, which is not suitable for mining sensor stream data. On the other hand, mining PLT needs an extra mapping mechanism for the sensors to a vector. Tanbeer et al. [18] has proposed a tree-based data structure called sensor pattern tree (SP-tree) to generate the set of all association rules from WSN data with one scan over the sensor database. Both PLT and SP-tree stored only the frequent sensor with a user given threshold. For this reason these tree are not suitable for finding associated sensor patterns.

Recently, researchers have focused on devising methods to mine user's *interest-based* frequent patterns to produce the desired result set in efficient manner by applying early pruning technique to reduce the size of resultant item sets. E. Miccinski et al. [11] proposed three alternative interestingness measures, called *any-confidence*, *all-confidence* and *bond* for mining associations for the first times. B. Liu et.al [12] analyzed contingency table for pruning and summarizing the discovered correlations. Later on, Y.K Lee et.al [13] used *all-confidence* to find correlated pattern. Algorithm proposed in [14] makes an effort to improve the performance of [13] by introducing items' support interval concept. Z. Zhou et al. [15] used a new interesting measure *corr-confidence* for rationally evaluating the correlation relationships.

TABLE I. A Sensor database (*SD*)

TS	Epoch
1	$s_1 s_2 s_3 s_4 s_7 s_8$
2	$s_1 s_5 s_6$
3	$s_2 s_5 s_6 s_7 s_8$
4	$s_1 s_2 s_4 s_7$
5	$s_1 s_2 s_4 s_5$
6	$s_1 s_2 s_3 s_4 s_7$

III. Associated Sensor Patterns Mining Problem in WSNs

Let $S = \{s_1, s_2, ..., s_p\}$ be a set of sensor in a particular wireless sensor network. We assume that the time is divided into equal-sized slots $t = \{t_1, ..., t_q\}$ such that $t_{j+1} - t_j = \lambda, j \in [1, q-1]$ where λ is the size of the each time slot. A set $P = \{s_1, s_2, ..., s_n\} \subseteq S$ 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 $Y \subseteq X$. Frequency of the pattern *Y* in *SD* is defined to be the number of epochs in *SD* that support it, i.e., $Freq(Y, SD) = |\{E(E_{ts}, X) | Y \subseteq X\}|$. Pattern *Y* is said to be a frequent pattern if $Freq(Y, SD) \geq min\_sup$, where *min_sup* is a user given *minimum support threshold* in percentage of *SD* size in number of epochs.

The interestingness measure *all-confidence* denoted by α of a pattern *Y* is defined as follows:

$$\alpha = \frac{Sup(Y)}{Max\_sensor\_Sup(Y)} \quad (1)$$

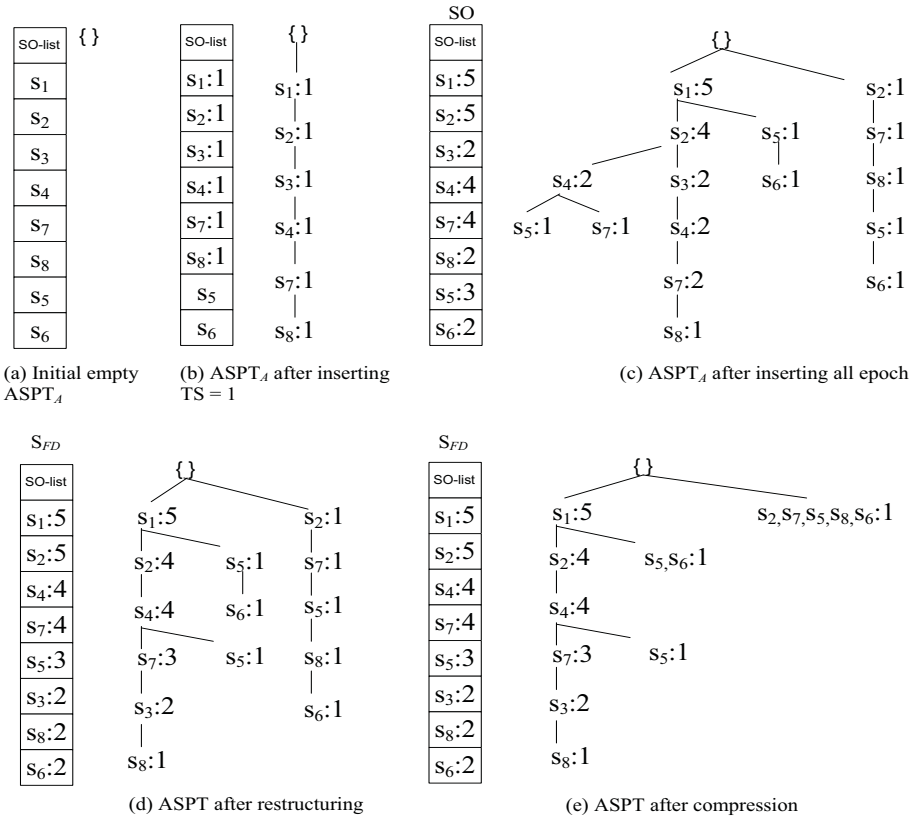

Fig. 1. ASPT Construction

Definition 1 (Associated Pattern): A pattern is called an associated pattern, if its *all-confidence* is greater than or equal to the given minimum *all-confidence* threshold.

Given a sensor database *SD*, $min\_sup(\delta)$ and $min\_all\_conf(\alpha)$ constraints, find the complete set of associated patterns in *SD* having the value no less than δ and α.

IV. Proposed ASPT Construction and Algorithm

In this section, at first we describe the construction of associated sensor pattern tree (ASPT). Then, we discuss the mining process in discovering associated sensor patterns from ASPT.

A. ASPT Construction

The ASPT construction has two phases: insertion phase and restructuring-compression phase. The step-by-step construction process of the ASPT based on the sensor database of Table I shown in Fig. 1(a-e). For the figure simplicity, we do not show the node traversal pointers in the tree.

For the insertion phase, ASPT arranges the sensors according to sensors' appearance order in the database and is built by inserting every epoch in database one after another. At this stage we call it ASPT$_A$, which simply maintains a sensor order-list (SO-list). The SO-list includes each distinct sensor found in all epochs in database according to their appearance and contains support value of each item in the database. Initially the ASPT is empty and starts construction with *null* root node shown in Fig. 1(a). Using *SD* in Table 1 as an example, the first epoch (i.e., TS =1) $\{s_1 s_2 s_3 s_4 s_7 s_8\}$ is inserted into the tree $< \{\} \rightarrow s_1 : 1 \rightarrow s_2 : 1 \rightarrow s_3 : 1 \rightarrow s_4 : 1 \rightarrow s_7 : 1 \rightarrow s_8 : 1 >$ as-it-is manner. Thus the first branch of the tree is constructed with s_1 as the initial node (just after root node) and s_8 as the last one as shown in Fig. 1(b). The support count entries for sensors s_1, s_2, s_3, s_4, s_7 and s_8 are also updated at the same time. Before inserting the second epoch, sensors of TS=2 are sorted from $\{s_1 s_5 s_6\}$ order to $\{s_1 s_2 s_3 s_4 s_7 s_8 s_5 s_6\}$ order to maintain the SO-list and then insert TS=2 into the tree. In this way, after adding all epochs (TS=6), a complete ASPT$_A$ shown in Fig. 1(c). Observe that each node in the ASPT$_A$ contains the occurrence frequency of the epochs, which represents the count of the pattern from the root to the node in the path. We call the final SO-list of the constructed ASPT$_A$ as SO. Here, the insertion phase ends and the restructuring-compression phase starts.

The purpose of the restructuring-compression phase is to achieve a highly compact ASPT which will utilize less memory and facilitate fast mining process. In this phase, we first sort the SO in frequency-descending order (S_{fd}) using merge sort and reorganize the tree structure according to S_{fd} order. For restructuring our ASPT, we use BSM (branch sorting method) proposed in [22]. 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

Algorithm 1 The ASP Algorithm

Input: SD, ISAO: Initial sensor appearance order, min_sup, min_all_conf
Output: Complete set of associated sensor patterns

Begin
$SO \leftarrow$ an SO-list arranged in ISAO
ASPT$_A \leftarrow$ a prefix-tree with null initialization
while (Not end of SD) **do**
  Scan an epoch from the current location in SD;
  Insert the scanned epoch into ASPT according to ISAO by following FP-tree construction method:
end while
Calculate S_{FD} from SO in frequency-descending order using merge-sort method;
for each branch in ASPT$_A$ **do**
  Sort the branch in S_{FD} using branch sorting method (BSM);
end for
for each branch in restructured ASPT$_A$ **do**
  Identify the *same support sensor node* in each branch and merge them to a single node
end for
while any mining request from the user **do**
  Input α and β from the user
  for sensor v from the bottom of SO-list **do**
    Create CPB tree CPB_v with SO-$list_v$ for sensor v
    Call Mining $(CPB_v, SO - list_v, v)$
  end for
end while
End

reinserts them into the tree. Fig. 1(d) shows the structure of ASPT after the restructuring operation. At this stage, we employ a simple but effective compression process that selects the *same support sensor nodes* in each branch and merge them into a single node. The final ASPT, after restructuring and compression is shown in Fig. 1(e).

Property 1: The support value of any node in ASPT is greater than or equal to the sum of total support value of its children.

Property 2: ASPT can be constructed in a single database scan.

Lemma 1: Given a sensor database *SD*, the complete set of all sensor projections of all epochs in the *SD* can be derived from its ASPT.

Proof: From the ASPT construction process, we can see that, all sensor projections of each epoch are mapped to only one path in the ASPT. For this reason, ASPT conserves a complete set of all sensor projections of each epoch for *SD* only once.

B. Mining Process

Let the *SD* presented at Table I, the constructed ASPT in Fig. 1(e), $min\_sup = 3$ and $min\_all\_conf = 0.55$. Like [10], we recursively mine the ASPT of decreasing size to generate associated patterns by creating conditional pattern-bases (PB) and the corresponding conditional trees

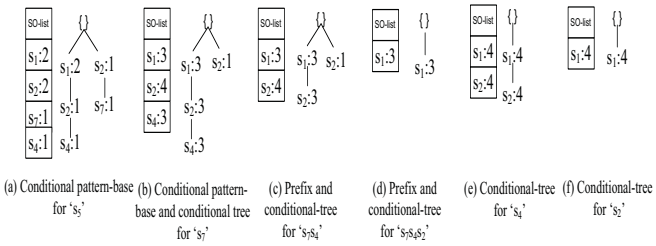

Fig. 2. Conditional pattern-base and conditional tree construction with the ASPT

(CT) without any additional database scan. We start building the conditional pattern-base and conditional trees from the sensor at the bottom of the S_{FD} list (Fig. 1(e)). The three bottom sensors s_6, s_8 and s_3 do not satisfy the min_sup threshold. Therefore, at first the conditional pattern-base tree of s_5 is created by taking all the branches prefixing the sensor s_5 as shown in Fig. 2(a). Sensor s_5 creates branches ($s_1s_2s_4$:1), (s_2s_7:1) and (s_1:1) where s_1:1 shares the prefix with $s_1s_2s_4$:1. The conditional-tree of s_5 is empty, because s_1, s_2, s_4 and s_7 do not satisfy the given min_sup and min_all_conf thresholds.

For the other sensors, the conditional pattern-base and the corresponding conditional trees are shown in Fig. 2(b-e). Now prefix and conditional-tree for s_7 is created in Fig. 2(b). Its conditional-tree contains two path $(s_1s_2s_4 : 3)$ and $(s_2 : 1)$ and the generated associated sensor patterns for s_7 are $s_1s_7 : 3$, $s_2s_7 : 4$, $s_4s_7 : 3$. The prefix and conditional-tree of associated sensor pattern s_4s_7 is created in Fig. 2(c). Patterns $s_1s_4s_7 : 3$ and $s_2s_4s_7 : 3$ are generated here. The prefix tree of patterns $s_2s_4s_7$ is shown in Fig. 2(d) and $s_1s_2s_4s_7$ is the generated pattern here. Now the prefix and conditional-tree for sensor s_4 is shown in Fig. 2(e). It has only one branch $(s_1s_2 : 4)$ and the generated associated sensor patterns are $s_1s_4 : 4$, $s_2s_4 : 4$ and $s_1s_2s_4 : 4$. The conditional pattern-base and conditional-tree for sensor s_2 is shown Fig. 2(f). The generated associated sensor pattern for s_2 is $s_1s_2 : 4$. Finally, the top-most item is s_1 and there is no associated sensor pattern for this. Algorithm 1 shows pseudo-code of ASP. The overall mining process for the given sensor database of Table I is shown in Table II.

With the above mining process, one can see that, for given min_sup and min_all_conf thresholds the complete set of associated sensor patterns can be generated from an ASPT constructed on a *SD*.

V. Experimental Results

To evaluate the performance of our proposed approach, we performed experiments on IBM synthetic dataset (*T10I4D100K*), real life dataset *BMS-POS* and *kosarak* from frequent itemset mining dataset repository [23]. Context and objects in these datasets are similar to the epochs and sensors in the terminology of this paper. The datasets are widely used in similar studies [17, 18]. We also used another dataset containing real WSN data from Intel Berkely Research Lab [24] which is widely used by many research community [16, 17, 21]. We utilized one datasets for historical periods of 10 days where 30 second is the slot size. Our programs are written in Microsoft Visual C++ and run with Windows 7

TABLE II. MINING THE ASPT BY CREATING CONDITIONAL (SUB-) PATTERN BASE

Sensor	Conditional Pattern-base	Conditional-tree	Associated sensor patterns
s_5	$\{(s_1 s_2 s_4 : 1), (s_2, s_7 : 1), (s_1 : 1)\}$	-	-
s_7	$\{(s_1 s_2 s_4 : 3), (s_2 : 1)\}$	$< s_1 : 3, s_2 : 4, s_4 : 3 >$	$s_1 s_7 : 3$, $s_2 s_7 : 4$, $s_4 s_7 : 3$, $s_1 s_4 s_7 : 3$, $s_2 s_4 s_7 : 3$, $s_1 s_2 s_4 s_7 : 3$
s_4	$\{(s1s2 : 4)\}$	$< s_1 s_2 : 4 >$	$s_1 s_4 : 4$, $s_2 s_4 : 4$, $s_1 s_2 s_4 : 4$
s_2	$\{s_1 : 4\}$	$< s_1 : 4 >$	$s_1 s_2 : 4$

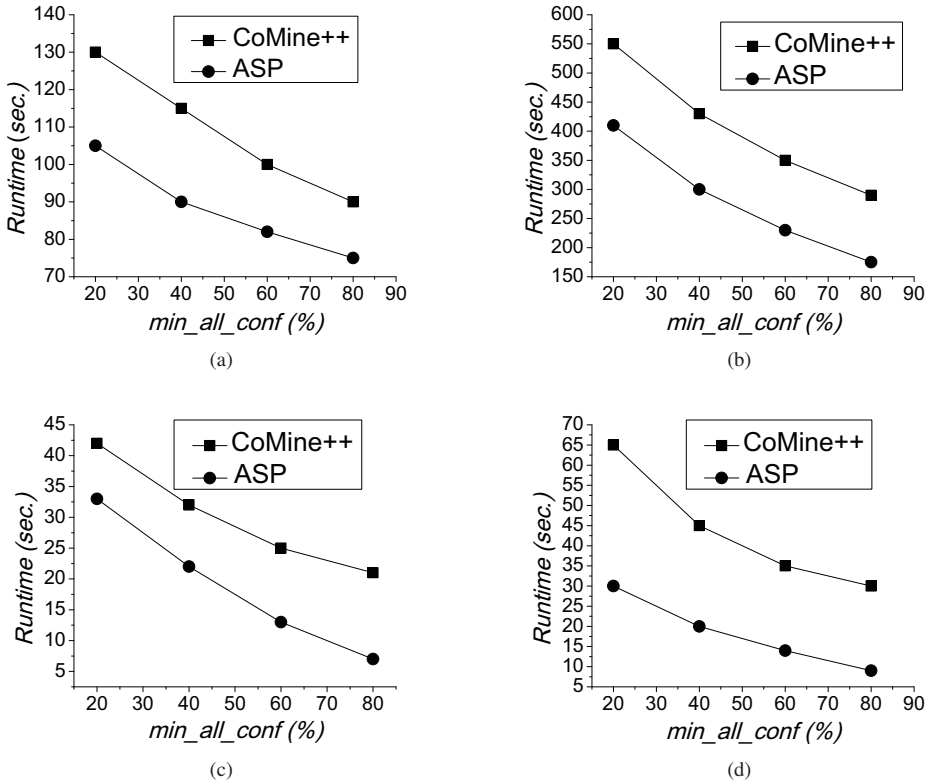

Fig. 3. Runtime comparison: ASP v/s CoMine++ when min_sup is fixed at (a) *T10I4D100K (10%)*, (b) *BMS-POS (3%)*, (c) *Kosarak (0.1%)* and (d) *10 Days data (30%)*

on a 2.66 GHz machine with 4GB of main memory. Runtime specifies the total execution time (i.e., CPU, I/Os) and includes tree construction, tree reconstruction and compression (for ASPT), and mining time.

Our experimental analysis is divided into three parts. First, we show its performance on mining the set of associated frequent patterns; second, we study the compactness of ASPT; and finally, we give the results to prove the scalability in mining associated frequent patterns by ASPT.

A. Execution time of the ASP

PLT [16] store only the frequent items with a given threshold. Therefore, it is not possible to find a set of associated frequent patterns from these tree. We compare our technique with CoMine++ [14], which proposed for mining correlated pattern mining for static transactional database. CoMine++ algorithm is not suitable for stream data mining due to scanning a database twice. In the first scan, it finds all single-element frequent patterns and in the second scan it performs the tree construction and mining operation. CoMine++ used FP-tree to represent the information of *SD* with respect to given user assigned min_sup and min_all_conf values. CoMine++ needs to design its tree structure again for every new user request because it does not maintain *build once and mine many* property. Firstly, the mining operation was performed by varying min_all_conf thresholds, where the min_sup values were fixed as 10%, 3%, 0.1% and 30% respectively for the above mentioned datasets. The results are represented in Fig. 3. From Fig. 3., it is shown that an increase in min_all_conf values decreases the runtime in both ASP and CoMine++. The reason is that, with the increase of the min_all_conf values, the associated patterns number is deceased, however for all case ASP is outperformer over CoMine++. Secondly, the mining operation is performed by varying min_sup thresholds, where the min_all_conf is fixed as 20% for all datasets. The results are shown in Fig. 4, where ASP always outperformed than CoMine++ over the entire supports of experiments.

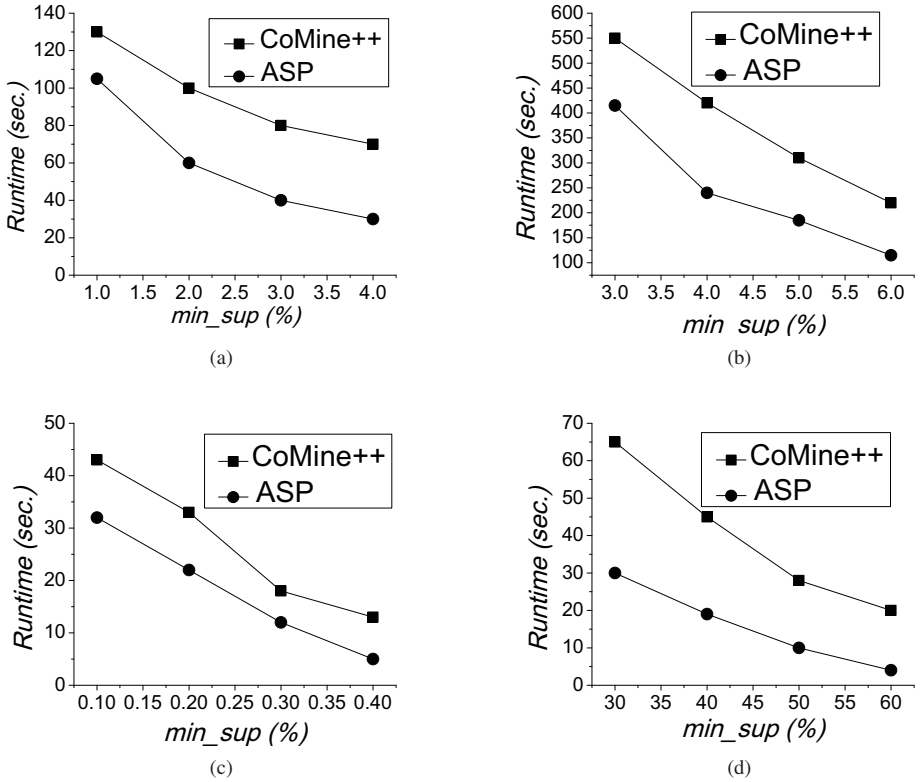

Fig. 4. Runtime comparison: ASP v/s CoMine++ when min_all_conf is fixed at 20% for all datasets (a) *T10I4D100K*, (b) *BMS-POS*, (c) *Kosarak* and (d) *10 Days data*

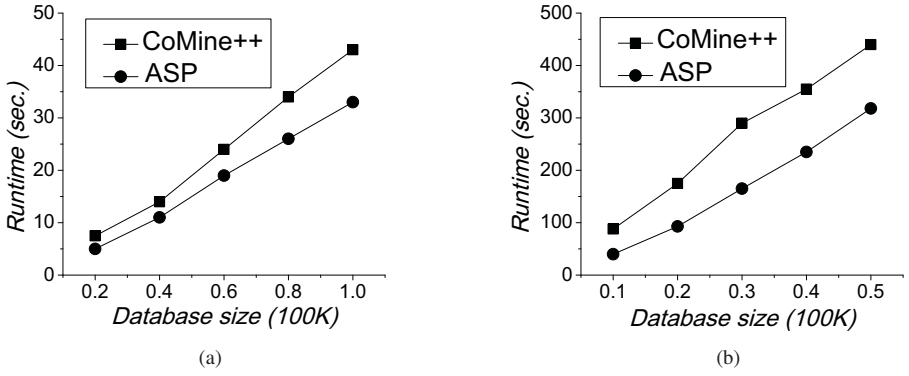

Fig. 5. Scalability comparison: ASP v/s CoMine++. (a) $Kosarak(\delta = 0.1\%, \alpha = 50\%)$ and (b) $BMS-POS(\delta = 3\%, \alpha = 40\%)$

B. Compactness of the ASPT

To show the compactness of ASPT we compared its size with PLT [16] and FP-tree [10] for given min_sup and min_all_conf thresholds. CoMine++ algorithm [14], used FP-tree for mining correlated pattern from transactional database. For PLT and FP-tree, we considered min_sup only, because they are support threshold-based tree structure. Their memory consumptions for different values of parameters using all the four data sets are shown in Table III. From Table III, we observe that, keeping the min_sup fixed, the memory usages of ASPT decreases with the increasing min_all_conf. From Table III, we also observe that an ASPT achieves compactness better than an PLT and FP-tree for all min_sup and min_all_conf.

C. Scalability of the ASP

For the scalability test, we used *kosarak* and *BMS-POS* datasets. *kosarak* dataset was divided into five portions, each of 0.2 million transactions. On the other hand, *BMS-POS* dataset also divided into five portions, each of 0.1 million transactions. The experimental results are presented in Fig. 5(a-b), where we fix min_sup 0.1%, min_all_conf 50% , and min_sup 3%, min_all_conf 40% respectively. Fig.5 shows the total execution time (including the ASPT

TABLE III. MEMORY COMPARISON AMONG ASPT, PLT AND FP-TREE

Dataset min_sup(shown as $\delta\%$)	Tree	min_all_conf (%)	Memory (MB) δ_1	δ_2	δ_3
T10I4D100K $\delta_1 = 1.0, \delta_2 = 2.0, \delta_3 = 3.0$	ASPT	20	6.10	3.10	0.21
		40	5.50	2.30	0.15
		60	4.60	1.20	0.09
	PLT	–	7.50	3.60	0.35
	FP-tree	–	8.35	4.66	0.50
BMS-POS $\delta_1 = 2, \delta_2 = 4, \delta_3 = 5$	ASPT	40	15.30	8.90	5.80
		50	12.40	7.20	3.40
		60	9.55	6.50	2.30
	PLT	–	20.30	15.5	9.60
	FP-tree	–	30.5	23.5	14.7
kosarak $\delta_1 = 0.2, \delta_2 = 0.3, \delta_3 = 0.4$	ASPT	20	670	540	330
		30	620	90	280
		40	580	350	250
	PLT	–	700	610	520
	FP-tree	–	3000	2000	1500
Intel data $\delta_1 = 30, \delta_2 = 40, \delta_3 = 50$	ASPT	20	1410	480	250
		40	1250	390	170
		60	1090	320	105
	PLT	–	1500	510	280
	FP-tree	–	3800	1250	700

constructing time and corresponding mining time) in the y-axis and the number of transactions in the x-axis. The ASP algorithm demonstrates stable result of about linear increase of execution time with respect to the size of the database.

VI. CONCLUSION

In this paper, we provide an efficient method for mining associated sensor patterns from WSNs data using a prefix tree called ASPT. We have used a pattern growth approach to avoid the level-wise candidate generation-and-test method. Our proposed, ASPT have the *build once and mine many* property and is highly suitable for interactive mining. This tree structure requires only one database scan to determine the complete set of associate sensor patterns. Extensive performance analysis shows that our tree structure is very efficient for associated sensor patterns mining and outperform the existing algorithm in both execution time and memory usage. Future research will explore ways to use the extracted knowledge to improved the performance and quality of services (QoS) of WSNs.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, 2002.

[2] W.B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho and Mark A. Perillo, "Middleware to support Sensor Network Applications," *IEEE Network*, PP. 6-14, 2004.

[3] F. Zhao and L.J. Guibas, "Wireless Sensor Networks: An Information Processing Approach," *Morgan Kaufmann publisher*, 2002.

[4] A. Boukerche, R.W. Pazzi, and R.B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," *Proc. 7th ACM Int. Symp. Model., Anal. Simul. Wireless Mobile Syst.*, pp. 157-164, 2004.

[5] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: An overview," *Advances in Knowledge Discovery and Data Mining*, pp. 1-34., 1996.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Mag.*, vol. 17, no. 3, pp. 37-54, 1996.

[7] P.-N. Tan, "Knowledge discovery from sensor data," *Sensors*, pp. 14-19, 2006.

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

[9] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. of the 20th VLDB Conf*, pp. 487-99, 1994.

[10] J. Han, J. Pei and Y. Yin, "Mining Frequent Pattern without Candidate Generation," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 1-12, 2000.

[11] E. Omiecinski, "Alternative interesting measures for mining associations," *IEEE Trans. on KDE*,vol. 15, no. 1, pp. 57-69, 2003.

[12] B. Liu, W. Hsu and Y. Ma, "Pruning and Summarizing the Discovered Association," *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 125-134, 1999.

[13] Y.K. Lee, W.Y. Kim, Y.D. Cai and J. Han, "CoMine: Efficient Mining of Correlated Patterns," *Proceedings of the Third IEEE International Conference on Data Mining*, 2003.

[14] R.U. Kiran and M. Kitsuregawa, "Efficient Discovery of Correlated Patterns in Transactional Databases Using Items' Support Intervals,"*DEXA 2102*, pp. 234-248, 2012.

[15] Z. Zhou, Z. Wu, C. Wang and Y. Feng, "Mining Both Associated and Correlated Patterns," *Computational Science–ICCS*, pp. 468-475, 2006.

[16] A. Boukerche and S.A. Samarah, "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.

[17] A. Boukerche and S.A. Samarah, "A New Representation Structure for Mining Association Rules from Wireless Sensor Networks," emph-Wireless Communications and Networking Conference, WCNC, pp. 2855-2860, 2007.

[18] S.K. Tanbeer, C.F. Ahmed, B.S. Jeong, "An Efficient Single-Pass Algorithm for Mining Association Rules from Wireless Sensor Networks," *IETE Technical Review*, Vol. 26, Issue 4, 2009.

[19] K.K. Loo, I. Tong and B. Kao, "Online Algorithms for Mining Interstream Associations from Large Sensor Networks," *Advances in Knowledge Discovery and Data Mining*, pp. 143-149, 2005.

[20] G.S Manku and R. Motwani, "Approximate frequency counts over data streams," *Proc. on VLDB*, pp. 346-357, 2002.

[21] K. Romer, "Distributed Mining of Spatio-Temporal Event Patterns in Sensor Networks," *EAWMS / DCOSS*, pp. 103-116, 2006.

[22] S.K. Tanbeer, C.F. Ahmed and B.S. Jeong, "Efficient Single-Pass frequent pattern mining using a prefix-tree," *Information Sciences (Elsevier)* vol.179, no. 5, pp. 559-583, 2009.

[23] http://fimi.cs.helsinki.fi/data/

[24] http://db.csail.mit.edu/labdata/labdata.html.