Enhanced SWASP Algorithm for Mining Associated Patterns from Wireless Sensor Networks Dataset

Surya S  
M. Tech. Student  
Department of Computer Science & Engineering  
NCERC Thrissur India

Mary Mareena P V  
Assistant Professor  
Department of Computer Science & Engineering  
NCERC Thrissur India

Dr. S Dhanabal  
Associate Professor  
Department of Computer Science & Engineering  
NCERC Thrissur India

Abstract

A Wireless Sensor Network are successfully deployed for various applications such as low cost area monitoring, environment monitoring, industrial and machine health monitoring, and military surveillance and they are spatially distributed autonomous sensors to monitor conditions such as physical and environmental. WSNs generate a large amount of data streams. Mining useful information from these data stream is a challenging task. Many algorithms have been proposed to extract the useful knowledge from sensor data and the widely used algorithm is Associated Sensor Pattern and compact tree structure, called Associated Sensor Pattern tree which uses pattern growth-based approach to generate all associated patterns with only one database scan over dataset. But when data stream flows through associated sensor pattern may fail to capture the significance of recent data. To overcome this limitation Associated Sensor Pattern tree is further enhanced to Sliding Window Associated Sensor Pattern tree by adopting sliding observation window and updating the tree structure accordingly and a mining algorithm Sliding Window Associated Sensor Pattern is used to mine recent associated patterns. But the limitation of Sliding Window Associated Sensor Pattern is that it takes too much time to mine associated patterns for large dataset. To enhance the performance of this algorithm, the dataset is partitioned into different sub parts and by using this partitioned dataset the Sliding Window Associated Sensor Pattern algorithm are run parallel by using multithread concept. And this enhanced algorithm reduces the total execution time and also increases the memory efficiency.

Keywords: Wireless Sensor Network, Sensor Data Stream, Behavioral Patterns, Data Mining, Knowledge Discovery

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are successfully used for various application and it is a promising and interesting research area for diverse monitoring [3] and detection application such as area monitoring, waste water monitoring, health monitoring and military surveillance. WSN consists of a large number of sensor nodes which communicates through wireless media to the central sink node and cooperatively works to monitor the environment. WSN generate a large amount of data streams. Extracting such data stream from WSN presents new challenges for data mining techniques. Data mining techniques, which are well established in the traditional database systems, have recently received a great deal of attention as promising tools to extract interesting knowledge from sensor data streams and these techniques have been used to extract useful knowledge from WSN data, through discovering relationships among the sensor nodes which are known as behavioral patterns. Behavioral pattern is used to identify missed reading events and better management of resources in a WSN. Discovering behavioral patterns from WSNs can be highly useful in applications that require a fine-grain monitoring of physical environments which may face critical situations like [8] fire, toxic gas leaks and explosion.

Using knowledge discovery in WSN one particular interest is to find behavioral patterns of sensor nodes, which are evolved from meta-data describing sensor behaviors. If events from sensors s1 and s2 are reported, then there is a 80 percent chance of receiving an event from sensor s3 and s4 within λ units of time, where 80 percent is the frequency of the rule. Generating association rules that have certain frequency needs to generate all the frequent patterns present in the data that meet a specific frequency value. The process of generating the association rules is straightforward after determining the frequent patterns. The rule scheme is dependent on a constraint termed minimum support threshold which specify minimum lower bound for the support of resulting association rules. It is possible to extract high value knowledge if the minimum support threshold is set high. By making the minimum support threshold low, an extremely large number of association rules are generated, most of which are non-informative. In this case, the valid correlation in the data objects gets covered among a huge pile of pointless rules.
Since WSN generates huge amount of data, it is necessary to use appropriate interestingness measure to find sensor behavioral patterns that have strong correlation among data. In response to this problem, a new type of sensor behavioral pattern called Associated Sensor Pattern. These behavioral patterns capture temporal correlation association like co-occurrence which are linked with such co-occurrences in the sensor data. To capture this type of behavioral patterns a compact tree structure, called Associated Sensor Pattern tree and a mining algorithm Associated Sensor Pattern are used, it requires only one scan over dataset and it use pattern growth-based approach to generate all associated patterns. Old information may lose significance for the current time, when data stream flows through as the Associated Sensor Pattern tree may fail to hold the information for longer period of time. To capture significance of recent data, Associated Sensor Pattern tree is further improved to Sliding Window Associated Sensor Pattern tree by adopting sliding observation window and the tree structure is updated accordingly. The limitation of Sliding Window Associated Sensor Pattern is that it takes too much time to mine associated patterns for large dataset. To enhance the performance of this algorithm, the dataset is partitioned into different sub parts and by using this partitioned dataset the Sliding Window Associated Sensor Pattern algorithm are run parallel by using multithread concept. And this enhanced algorithm reduces the execution time and also increases the memory efficiency.

II. RELATED WORK

Wireless sensor network is successfully deployed in diverse monitoring and detection applications. In these applications. WSNs generate a large amount of data in the form of streams. Such data stream from WSN can be mined to extract useful knowledge but it is a challenging task. Discovering behavioral patterns that mean associated patterns from wireless sensor networks dataset can be highly useful in applications that require a fine-grain monitoring of physical environments such as transportation networks, battle field. Behavioral patterns can also be used to predict the source of future events. Because of these various applications, various approaches have been used to extract useful information. The approaches which are suggested earlier in order to extract useful information are as follows:

A. FP-tree

J.Han et al. [9] uses a novel data structure, frequent-pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree based mining method, FP-growth. There are several advantages of FP-growth over other approaches. First, it constructs a highly compact FP-tree, which is usually substantially smaller than the original database and thus saves the costly database scans in the subsequent mining processes. Second, it applies a pattern growth method which avoids costly candidate generation and test by successively concatenating frequent 1-itemset found in the (conditional) FP-trees. This ensures that it never generates any combinations of new candidate sets which are not in the database because the itemset in any transaction is always encoded in the corresponding path of the FP-trees. The major operations of mining are count accumulation and prefix path count adjustment, which are usually much less costly than candidate generation and pattern matching operations performed in most Apriori-like algorithms [10]. Third, it applies a partitioning-based divide-and-conquer method which dramatically reduces the size of the subsequent conditional pattern bases and conditional FP-tree. However, the requirement of two database scans for this kind of trees is not suitable for generating association rules from WSN data.

B. Positional Lexicographic tree

Azzedine Boukerche et al. [2] uses a Positional Lexicographic Tree (PLT) is used to store a sensor’s event detecting status. Its mining process follows the pattern growth approach. The process is same for all of the sensors presented in the PLT structure. Its mining starts with the sensor having maximum rank by generating the frequent patterns from its PLT in a recursive way. Unlike frequent pattern tree, the partitioning mechanism used in PLT for particular pattern it makes it easy to locate the conditional vectors instead of following the nodes link as in frequent pattern tree. As opposed to FP-tree, in PLT there is no need to maintain an entire structure in main memory. Likewise, there are various issues that makes PLT outperforms the FP-tree. The disadvantage of PLT in mining is that it needs a mapping mechanism to transform sensor data to a position vector and also the requirement of two database scans for this kind of tree is not suitable for mining association rules from WSN data.

C. CanTree

Leung et al. [4] uses a novel tree structure called Canonical-order tree (CanTree). From transactional database it captures the content with a single scan for interactive and incremental frequent pattern mining. By using some canonical order, it orders the tree accordingly. By exploiting its nice properties, when database transactions are inserted, deleted or modified, the CanTree can be easily maintained. For example, the CanTree does not require adjustment, merging, or splitting of tree nodes during maintenance. For incremental updating, no reconstruction of a new tree or repeated scanning of the entire updated database is needed. However, for the frequency independent canonical-order item insertion, CanTree achieves less compactness and thus results in poor mining performance than frequent Pattern tree.
D. **Compact Pattern tree**

S.K.Tanbeer et al. [5] uses a novel tree structure, called CP-tree (Compact Pattern tree) that captures database information with one scan in the insertion phase and provides the same mining performance as the FP-growth method in the restructuring phase. At runtime dynamic tree restructuring concept is used by CP-tree to produce a highly compact frequency-descending tree structure. The CP-tree overcomes the drawback of both FP-tree and CanTree and offers a highly competent frequent pattern mining solution for incremental and interactive mining. On the Prefix-tree, the CP-tree ensures FP-growth mining performance for an entire database in memory efficient manner. However, it is not possible to find associated patterns it only mines frequent patterns.

E. **CoMine Algorithm**

Y.K.Lee et al. [6] used all-confidence to find correlated patterns as it satisfies both null-invariance and downward closure property. CoMine uses pattern-growth technique to discover the complete set of correlated patterns that satisfy the user-defined minimum support and minimum all-confidence constraints. The CPB of the suffix pattern in CoMine represents the set of complete prefix paths in FP-tree co-occurring with itself. Thus, CoMine implicitly assumes that the suffix pattern can concatenate with all items in its prefix paths to generate correlated patterns of higher-order. Such an assumption can make some performance problems in CoMine. Although CoMine algorithms are efficient for transactional database, it is not suitable for sensor data stream because of multiple scan requirement of same dataset.

F. **CoMine++ Algorithm**

R.U.Kiran et al. [7] in order to improve the performance a novel concept called items support interval is used and an algorithm CoMine++. To discover correlated patterns effectively CoMine++ is used. The main contributions of CoMine++ are as follows. A novel concept known as items’ support intervals is used. It states that, to form correlated patterns of higher-order, an item having supports can combine with only those items having supports within a specific interval. Unlike CoMine, the CPB of the suffix item in CoMine++ represents the set of partial prefix-paths that means involving only those items that have support within the support interval of suffix item in FP-tree co-occurring with itself. Using the prior knowledge regarding the construction and mining of FP-tree, CoMine++ uses a novel pruning technique to construct the CPB of the suffix item effectively. Although CoMine++ algorithms are efficient for transactional database, it is not suitable for sensor data stream because of multiple scan requirement of same dataset.

G. **ASP and SWASP**

M.M.Rashid et al. [1] uses a new type of behavioral pattern called associated sensor patterns which capture temporal correlations as well as association like co-occurrences as well which are linked with such type of co-occurrences. To capture this type of patterns a mining algorithm called Associated Sensor Pattern and a compact tree structure, associated Sensor Pattern tree is used and to generate all associated patterns with only one scan over database which use pattern growth based approach. Associated Sensor Patterns capture association-like co-occurrences as well as temporal correlations which are linked with such co-occurrences. In ASP when data stream flows, old information may lose significance for the current time. ASP-tree is further enhanced to SWASP-tree by adopting sliding observation window to capture importance of recent data. The limitation of SWASP is that it takes too much time to mine associated patterns for large dataset.

In the related works, the limitation of FP-tree, PLT-tree, CoMine algorithm, CoMine++ Algorithm to mine associated patterns from sensor stream data is that the requirement of double database scan. However, the CP-tree only mines frequent patterns, so it is not possible to mine associated patterns and for CanTree, in frequency independent canonical-order item insertion, CanTree achieves less compactness and thus results in poor mining performance than frequent Pattern tree. The limitation of SWASP is that it takes too much time to mine associated patterns for large dataset.

**III. PROPOSED WORK**

The limitation of the Sliding Window Associated Sensor Pattern is that it takes too much time to mine associated patterns for large dataset. To enhance the performance of this algorithm, the dataset is partitioned into different types and the Sliding Window Associated Sensor Pattern algorithm are run parallel by using this partitioned dataset by using multithread concept. And this enhanced algorithm reduces the execution time and also increases the memory efficiency. Multi-threading is a technique by which, at different stages of execution, a single set of code can be used by several processors. And in other words it is the single-processor way of running, simultaneously, more than one task. As running of multiple processes, it works the same way, through a timer interrupt, saving its machine state by suspends one and replacing that by the previously saved state of another, the only difference being that two threads share the same virtual memory space, which is of the same process, making the task switch much more efficient.

By using this method, it is possible to mine associated patterns better than that of the existing algorithm. In SWASP, to mine associated patterns it has to perform various steps that means it contains insertion phase, restructuring compression-phase, after that it has to merge the same support sensor nodes in each branch into single node. After that based on the user specified minimum support and minimum all confidence it starts to mine patterns which have thresholds higher than that of the user.
Enhanced SWASP Algorithm for Mining Associated Patterns from Wireless Sensor Networks Dataset

(IJIRST/ Volume 3 / Issue 02/037)

specified minimum support and minimum all confidence thresholds. Then it creates conditional pattern base and the corresponding conditional tree without any additional database scan and then mine associated pattern from this. In Sensor Data Stream mining, the mining is based on sliding window, in that a window consists of multiple times of non-overlapping batches and non-empty set of epochs is called batch. The SWASP-tree, captures useful knowledge from the stream content where batch-by-batch support information for the current window and sensor-id is maintained by each node in the tree. The construction of SWASP-tree proceeds window by window. In SWASP, there involves various steps, to mine associated patterns, but to run these various steps of the SWASP algorithm, parallel, by using multithread concept is not possible.

This is because in the insertion phase of the SWASP, it arranges the sensors according to sensors appearance order in the database and is built by inserting every epoch in the database one after another into it and it contains the support value of each item in the database. In the restructuring compression phase of the SWASP, the purpose of restructuring-compression phase is to achieve a highly compact tree, which will utilize less memory and facilitate fast mining process. In this restructuring-compression phase, it first sort the sensor order in frequency-descending order using merge sort and reorganize the tree structure according to frequency-descending order. To reconstruct the tree, it uses a tree restructuring technique called branch sorting method is used and it uses merge sort to sort every path of the tree structure. This approach first removes unsorted paths, then sorts all paths and reinserts them into the tree. At this stage, a simple but effective compression technique like CanTries [3] that selects the same support sensor nodes in each branch and merge them into a single node. In SWASP, to capture the significance of recent data old information have to be deleted and the new information have to be added. Because of this step it takes more time to mine associated patterns. After this procedure the SWASP has the same insertion phase and the restructuring compression phase, after all these steps only, it can mine associated patterns.

So to reduce the execution time of SWASP algorithm, it can be enhanced by partitioning the large dataset into sub parts, and the SWASP algorithm is run by using each partitioned dataset in parallel by using multithread concept. By using this method, it is possible to mine associated patterns better than that of the SWASP algorithm. The flow diagram of the enhanced SWASP algorithm is given below. In that the dataset is denoted by \( D_{(1)} \) and it is partitioned into different sub dataset \( D_{(1,2)}, D_{(1,3)}, \ldots D_{(1,n)} \) and each dataset is chosen by SWASP algorithm to run parallel by using multithread concept Multi-threading is a technique by which at different stages of execution, a single set of code can be used by several processors. And in other words it is the single-processor way of running, simultaneously, more than one task. As running of multiple processes, it works the same way, through a timer interrupt, saving its machine state by suspends one and replacing that by the previously saved state of another, the only difference being that two threads share the same virtual memory space, which is of the same process, making the task switch much more efficient.

![Fig 1: Data Flow diagram](image-url)

A thread is a flow of execution through the process code, system registers, with its own program counter, and stack. A thread is also called a light weight process. Through parallelism threads provide a way to improve application performance. A software approach is used by threads to improving performance of operating system by reducing the overhead, thread is equivalent to a classical process. Context switching time can be easily minimized by thread. By the use of threads, it is possible to provides concurrency within a process. It provides efficient communication. To create and context switch threads, it is more economical. So the SWASP algorithm is run by using partitioned dataset in parallel by using multithread concept. The different result of the SWASP algorithm which is obtained by using different sub dataset is merged. From that the final result, that means associated pattern is obtained. The enhanced SWASP algorithm reduced the execution time and also increases the memory efficiency of SWASP algorithm.
IV. ANALYSIS

In this for evaluating the performance of SWASP and Enhanced SWASP, sensor dataset is used. A sensor database, is defined to be a set of epochs, where each epochs is a tuple $E(E_{\text{ts}}, Y)$, such that $Y$ is a pattern of the event detecting sensors that report events within the same timeslot and $E_{\text{ts}}$ is the epoch’s time slot. Runtime specifies the total execution time (i.e., CPU, I/Os) and includes tree construction, tree reconstruction and compression (for ASP/SWASP-tree), and mining time. But the limitation of Sliding Window Associated Sensor Pattern is that it takes too much time to mine associated patterns for large dataset.

To enhance the performance of this algorithm, the dataset is partitioned into different sub parts and by using this partitioned dataset the Sliding Window Associated Sensor Pattern algorithm are run parallel by using multithread concept. And this enhanced algorithm reduces the execution time and also increases the memory efficiency.

To know the performance analysis of both SWASP and Enhanced SWASP in terms of execution time and memory sensor dataset is used. In Fig. 2 for each size of the dataset, the dataset is partitioned into different sub parts, and for these partitioned dataset the SWASP algorithm is run parallel by using multithread concept, and the final result is merged and obtained the result, and for this dataset the consumption of memory is noted and is denoted as Bytes. Likewise for each size of the dataset, the same procedure is taken place. As the size of the dataset increase, the memory consumption of both SWASP and Enhanced SWASP also increase, but it is clear from the graph that memory consumption of Enhanced SWASP is less than that of SWASP.

![Fig. 2: Performance Analysis in terms of memory](image)

To know the performance analysis of both SWASP and Enhanced SWASP, in terms of memory sensor dataset is used. In Fig. 3, for each size of the dataset, the dataset is partitioned into different sub parts, and for each size of the dataset, the dataset is partitioned into different sub parts, and for these partitioned dataset, the SWASP algorithm is run parallel by using multithread concept, and the final result is merged and obtained the result. And for this dataset, the execution time of both SWASP and Enhanced SWASP is noted. Likewise, for each size of the dataset, the same procedure is taken place. As the size of the dataset increases, the execution time of both SWASP and Enhanced SWASP also increase, but it is clear from the graph that execution time of Enhanced SWASP is less than that of SWASP.

![Fig. 3: Performance Analysis in terms of Execution time](image)
V. Conclusion

Because of various application the importance of wireless sensor network is increasing day by day. There are various data streams in WSN and extracting useful information from this is a critical issue. So by using ASP-tree and a mining algorithm ASP it can mine associated sensor pattern efficiently but for more data streams this tree is not good enough. So to overcome this sliding window based ASP-tree is introduced and it is called SWASP-tree and a mining algorithm SWASP is used to capture the significance of recent data. But the limitation of Sliding Window Associated Sensor Pattern is that it takes too much time to mine associated patterns for large dataset. To enhance the performance of this algorithm, the dataset is partitioned into different sub parts and by using this partitioned dataset the Sliding Window Associated Sensor Pattern algorithm are run parallel by using multithread concept. And this enhanced algorithm reduces the total execution time and also increases the memory efficiency.

REFERENCES


