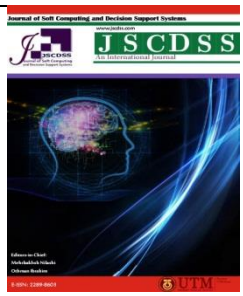
An International Journal
JSCDSS
http://www.jscdss.com
Vol.7 No.3 June 2020: 32-39
Article history:
Accepted 7 May 2020
Published online 7 May 2020

Journal of Soft Computing and Decision Support Systems



E-ISSN: 2289-8603

Distributed Frequent Itemset Mining with Bitwise Method and Using the Gossip-Based Protocol

Hoda Rafieipour [a], Azadeh Abdollah Zadeh [b,*], Mehrdad Mirzaei [c]

[a] Department of Computer Science, Memorial University of Newfoundland, NF, Canada, A1C 5S7
[b] Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, FL, USA, 33431
[c] Department of Computer Science, University at Albany, NY, USA, 12222

*** Corresponding author email address**: aabdollahzad2016@fau.edu

Abstract

Nowadays, distributed systems are prevalent and practical in network environments. In distributed systems, pattern recognition help to extract information from network nodes. Meanwhile, data mining in such systems needs resource consideration in terms of storage and computational time. The primary requirement of these systems is a scalable mechanism to distribute the tasks on several databases. Moreover, to do a centralized process, relocating data from all nodes or partial nodes to a central node has confidential risks and traffic overhead. Therefore, distributed data mining in distributed environments needs systematic and structural techniques. In this paper, we propose a new algorithm to extract frequent itemsets in Wireless Sensor Networks. Through this algorithm, nodes frequent local itemsets are obtained with a Bitwise approach, and nodes are classified into clusters by using the Low Energy-Adaptive Clustering Hierarchy (LEACH) algorithm. Connecting the head cluster is performed by a Gossip-based protocol to achieve the values of global support, and it finally resulted in the extraction of frequent itemsets. The proposed algorithm has been simulated in various scenarios using Java software, and algorithm efficiency is evaluated in terms of execution time and average accuracy. Our algorithm is compared with a Gossip-based algorithm, and then some improvements in execution time have been presented.

Keywords: Frequent Itemset mining, distributed data mining, Gossip-based protocol, Bitwise approach

1. Introduction

Recently, different types of sensors are widely used, such as presser sensors, flow sensors, fluid sensors, and airflow sensors. Mohamed et al. (2019) utilized piezoelectric sensors for detecting external loading in structures fabricated by additive manufacturing. One piezoelectric element employed for exciting the part and another piezoelectric sensor is implemented for monitoring the response of the structure to different ultrasonic wave excitation. This method of Structural Health Monitoring (SHM) is called Surface Response to Excitation (SuRE) and has shown promising results for fault detection (Mohamed et al., 2019).

In structural health monitoring, the structure is monitored in predefined periods, and the results will be sent to a remote server for extra processing. Recently, compressive sampling is introduced as an efficient, fast, and linear method of data sampling. The simulation results show that this technique in recovery can highly improve the quality, while sensors can compress the signal in minimal size (Surakanti et al., 2019). In the case of mobile devices, it must be considered that the time for generating compressed samples is a crucial factor. Besides, the fact must be noted that data should be sent to the practitioner as soon as possible. On the other hand, the wearable ECG recorders that have restrained power, and may only be intelligent in doing simple algorithms. Izadi and his colleagues (Izadi et al., 2020) proposed a system architecture that can produce compressed ECG samples, in a linear method by CR 75%. They used sparsity of the ECG signal and suggested an approach based on Compressed Sensing (CS) that can compress ECG samples in real-time. In such cases, finding a DS attack is equivalent to finding the server, which has a frequency higher than a threshold.

The most crucial goal of distributed data mining is to obtain the result similar to the one received by centralized data mining so that it would not be necessary to transfer the original data from their primary storage location. One of the applications of identifying and extracting frequent itemsets in distributed networks is Denial of Service (DoS) attacks detection in the distributed model. In the distributed Denial of Service Attack, target attacks are made from different places in the network. In these attacks, several malicious nodes continuously send a great deal of traffic to the victim node, which is usually a server. Therefore, the

total bandwidth is occupied, and authorized nodes do not have access to the desired service. In this paper, the extraction of frequent itemsets in Wireless Sensor Networks is implemented to divide the problem into two sub-problems.

First, we need to calculate the local frequency of itemsets. Then a bottom-up, horizontal Bitwise method is applied to extract the frequent itemsets, which are the centralized extraction of frequent itemsets, after reaching the global min-sups. The transaction database is distributed among sensor nodes in the network in a non-overlapping way, and each node attempts to count the *K* frequency of itemsets locally and by transforming the dataset to Bit matrix and generating a horizontal search tree with a depth traversal from root to node.

1-member itemset to *k*-member itemset (*1*-itemset, *2*-itemset, …, *K*-itemset) for the second sub-problem means that the LEACH-clustering algorithm was used to connect the nodes, and they are classified in several clusters with a hierarchical structure. In each cluster, each node connects to the head cluster, and each stable head cluster is connected to the others by a Gossip-based protocol. The nodes of each cluster are connected to their head cluster, and a Gossip-based protocol is used to connect head clusters, which are stable and quickly convergent.

The rest of the paper includes the following sections. In Section 2, we collect some works of distributed frequent pattern mining algorithms. Section 3 contains a definition of the problem, and prerequisites are dealt with in Section 3. In Section 4, we propose our algorithm. At the end of the paper, the results of tests and algorithm evaluation are presented in Section 5. Finally, in Section 6, conclusion and future works are presented.

2. Related works

In this section, we discuss distributed patterns mining algorithms, which are mainly based on Apriori and FP-Growth algorithms. As one of the earliest distributed Apriori, we can consider Count Distribution in (Agrawal and Shafer, 1996). In the Count Distribution, Each node sends its candidate items to other nodes with their supporting values. Therefore, each node can extract frequent global itemset. The connection cost of the algorithm depends on the length of the largest frequent itemset. Each node sends its data to *n-1* other ones in its multi-cast operation. Therefore, connections among nodes will cause significant overhead.

DDM algorithm (Park and Kargupta, 2002) is based on Apriori. In the architecture of this method, there is no shared memory. For each node, itemsets support is calculated individually and locally, and then a distributed decision protocol is run to specify that the frequent global itemsets be executed. The main idea of DDM is to determine whether an itemset is frequent before collecting the support values of itemsets from all nodes. This algorithm is acceptable in terms of cost. It is also scalable; however, the disadvantage is its runtime.

The D-Sampling algorithm is proposed by Schuster et al. (2005). To achieve the D-sampling method, a centralized version of a sampling method and DDM algorithm are mixed. A centralized database with distribution at runtime is the structure of the D-sampling algorithm. Each node is responsible for an itemset, and the algorithm to the memory loads a sample from the database. Each node distributes the sample so that the database is vertically divided. Then a version of the DDM algorithm is executed on the datasets of each node. After generating a frequent itemset, we scan the entire dataset to obtain support for this itemset. Since the algorithm is based on sampling, it is probable to lose some of the frequent itemsets.

An algorithm named MLFPT is presented by (Zaïane et al., 2001). A shared-memory structure is established in this algorithm. MLFPT, like centralized FP-Growth, does not produce a candidate for frequent itemsets, and constructing the frequent pattern tree (FP-Tree) requires only two scans of the database. It cannot be used for load balancing in other algorithms.

Tanbeer et al. (2009) introduced a specific type of parallel pattern tree (PP-Tree) in which the overhead of connections among the nodes and I/O costs are decreased. It happens since the algorithm receives the database contents in a single scan. The algorithm is executed locally in each part, and the frequent global patterns are locally generated. Then, they are merged in the final stage. The main idea in PP-Tree is to receive the contents of the database with one scan of a specific sequence and organizing the tree structure in descending order of global item's frequency (similar to FP-Tree). In this method, each PP-Tree is locally scanned so that the patterns, which potentially exist in all global data, can be extracted without the internal connections among nodes. Finally, the frequent global patterns are extracted with some small sequential paces.

An association rule mining algorithm with the goal of fast converging the ultimate answer is presented by Wolff and Schuster (2004). This algorithm consists of two approximately separate parts. Each node executes a sequential ARM algorithm on its local database and keeps the result. Moreover, each node participates in a majority voting protocol so that all nodes, which are accessible through each other, converge on the correct result.

In the ZigZag algorithm (Otey et al., 2003), each node generates its maximal frequent itemset first, and then global Maximal Frequent Itemsets (MFI) will be generated. Using MFI, all frequent itemsets will be calculated with one scan of each local dataset and without generating an infrequent candidate. In the FDM algorithm (Cheung et al., 1996), each node calculates the support of its itemsets, and all infrequent local itemsets will be pruned. After local pruning, each node sends a broadcast message including all of its local itemsets to other nodes and requests the support of these itemsets from other nodes. Accordingly, each node decides whether the itemsets are globally frequent or not. This process will be repeated until there is not a frequent itemset. An algorithm in which ring topology is generated as the main structure of a peer-to-peer network is presented in (Guan and Ip, 2007). Assume that *n* huge databases are

An International Journal
JSCDSS
E-ISSN: 2289-8603

distributed in the network, and each of these databases, which we call local database (D_i), is placed in these nodes. The advantage of this topology is the perfect scalability.

Network communications are performed asynchronously. Each node generates its local *k-item* candidate sets, C_i^k. Then, the node combines it in a data transfer network so that locally combined candidate *k-item* set (B_i^k) is obtained. Finally, the *k-item* frequent set (l_k) is generated in the n^{th} node. It is used as a seed to scan and generate local candidates *(k+1)-item*, C_i^{k+1}, in the next round.

This method reduces the generation of unnecessary data on each node as much as possible. Besides, all the processes of this algorithm are performed on each node in a parallel way. The disadvantage of this method is that it is focused only on networks with ring topology.

In a method called ODAM proposed by Ashra et al. (2004), in each site, *1*-itemset is calculated and broadcasted. Then, global *1*-itemsets are discovered. Additionally, second candidate itemsets are produced by the algorithm, and its support count is calculated as well.

Simultaneously, infrequent itemsets are eliminated. ODAM generates global *2*-itemsets, and the iteration through the main memory transaction is performed. The efficiency of the ODAM algorithm compare to CD and FDM is higher. As the number of sites is increasing, the size of the exchange message is increased. The number of exchanging messages is decreased than FDM. In the ODAM algorithm, the comparison numbers for generating the support count are minimized.

Lin et al. (2019) Provided a method called Fast Distributed Mining in Changing Bandwidth (FDCNB-mining), which takes into account the variation of network bandwidth. One of the best characteristics of the method is that it provides an appealing load balancing and efficiency. The structure of the work is based on CARM (K. W. Lin and Lo, 2013), along with adding novel nodes as monitors in each cluster to empower the system. The FDCNB-mining is tested with six different bandwidths for computing nodes. Results show that the FDCNB-mining algorithm in environments with variable bandwidths is superior to previous methods like the FLR-mining algorithm (Lin and Chung, 2015).

(Sumalatha and Subramanyam, 2020) Presented a Distributed High Utility Time Interval Sequential Pattern mining (DHUTISP) method, and they work on the big data by Map Reduce platform.

The notable feature of this work is proposing an exquisite Time Interval Utility List (TIUL) data structure. Having TIUL, the utility of the pattern is calculated, and two additional utilities like Utility Upper Bound (RUUB) and Co-occurrence Utility Upper Bound (CUUB) help the method to differentiate between the proper and useless patterns and minimize the search space. The time interval assists the algorithm setting a time interval for previous items to produce practical sequential patterns.

The DHUTISP method works better than top distributed and non-distributed methods. This method is superior to other current methods.

3. Fundamentals

3.1. Problem Definition

WSN network contains *n* nodes and a dataset named *A*, including *m* frequent itemsets. Each node ($i \in 1, 2, \ldots, n$) has a local view of these itemsets. Each node has a local itemset ($A_i \subseteq A$). Each item ($j \in A_i$) has a local value, which is indicated with V_i, in the node *i*. This value is equal to the summation of regional nodes. The objective is to find items that have local values higher than min-sup. We assume that min-sup is equal to *K*. Then, we consider an item frequent if the Eq. (1) condition is met:

$$Frequent\ items(A,K)=\{items \mid item \in V_j > K\} \quad (1)$$

In the proposed algorithm for node connections, we cluster the nodes by using Low Energy-Adaptive Clustering Hierarchy (LEACH) algorithm, and local information is delivered to the appropriate head cluster. A monotonous Gossip-based protocol is used in head clusters, which are considered the most basic gossip model.

3.2. Prerequisites

3.2. 1. LEACH Algorithm

LEACH algorithm is presented by Heinzelman et al. (2000). This protocol is the first dynamic clustering protocol that considers the need for WSN networks specially and uses homogeneous static nodes that are haphazardly distributed.

Generally, this protocol is a hierarchical, distributed, and one-step protocol. LEACH exploration contains different periods with a start-up phase plus a steady-state phase. The time in which nodes build a cluster called the start-up phase. A random number as *T*, which is between zero and one, is randomly selected by each node. Based on the value of *T* value, if the *T* is less than the threshold *T(i)* in the Eq. (2) the nodes are selected to be the cluster head.

$$T(i)=\begin{cases}\dfrac{p}{1-p*\left(r \bmod \frac{1}{p}\right)} & if\ i \in G \\ 0 & Otherwise\end{cases} \quad (2)$$

In the Eq. (2), *p* is the percentage that is required from the head cluster nodes in the sensors population, and r is the time to select the head cluster. There are some nodes without a head cluster, and those nodes are *G*. Then, all head clusters post an announcement to those nodes that are not head clusters to specify a cluster membership. Each node selects its cluster in a way, which it can communicate, with the head cluster with the least energy. Based on the head cluster announcement signal, if the node receives a strong signal of a head cluster, it is selected to be the head cluster member. Afterward, the chosen head cluster receives a message from each node that notifies the head cluster of the node membership of the cluster.

An International Journal
JSCDSS
E-ISSN: 2289-8603

Subsequently, the steady-state is started with the network performance break down into a couple of frames. For each node, a period named times slice is designated to let the node send its data to the head cluster. A communication mechanism is called Time-division multiple accesses (TDMA) is provided in which the head clusters build a communicative scheduler.

In this method, the period for sending data depends on the number of the cluster node. While all the nodes sending are finished, the head cluster integrates the node data in a package. In our proposed algorithm, the data are sent via head clusters with the Gossip-based method, and we are free of the centralized station for transmitting data. The head clusters send the data in a predefined period. At the end of the present time, the head cluster role is given to another node. This mechanism helps the load balancing since the flow of data is not just on one node, and it spread among the nodes.

Some advantages of this protocol are pointed out in the following:

i. This protocol is a random one. It means that in each time slices, a certain number of nodes are selected to be a head cluster, the algorithm randomly selects the nodes to be a head cluster, and the role is not steady for each node.
ii. This protocol is a comparative one. In this protocol, the algorithm selects the head cluster by chance, and the one is the head cluster in the current period is not the next time head cluster candidate. In such a manner, all the nodes of a cluster have the chance to be a head cluster.
iii. The proposed protocol is a self-organized algorithm. In this protocol, cluster construction is free of helping any other agent, and this fact supports the scalability of the protocol.
iv. MAC (Medium Access Control) protocol is used in this protocol, which lets the nodes to be on and off wave radio so that the energy of nodes is saved.
v. In the proposed algorithm, head clusters integrate the data from cluster nodes before transferring the data to peers, and it helps to minimize the size of data.

3.2. 2. Gossip-Based Protocol

The Gossip-based protocol is a type of communication method, which is inspired by the gossip distribution model in social networks. Modern distributed systems often use the Gossip-based protocol to handle the problems, which are difficult to be solved by the other solutions because of structure inappropriate or vastness of the system, or the problems in which gossip is the only efficient way.

Due to scalability and rapid convergence, this protocol is applicable in distributed and large networks. In an ordinary Gossip-based protocol, nodes in specific time transfer their data to other nodes that are randomly selected. It means that one node calls another node; therefore, sender and receiver nodes form a channel to transfer the data. This protocol has the power to distribute the information stably. The central protocol core is intermittent and two-by-two.

In this protocol, information size is limited, and the algorithm selects the nodes randomly. It selects the nodes among the full set of nodes or a smaller set named the neighbors set. The number of periods required for the distribution in the entire network is in the order of *log (n)*.

4. Distributed frequent itemset mining

In this section, we explain our proposed algorithm and our method of mining the frequent patterns in a distributed environment.

The transaction database is divided into *n* parts (*n* is the number of sensor nodes in the network), and each node is responsible for processing one part. The proposed algorithm is performed in two phases. The first phase is to calculate the local frequency, while the second phase is the aggregated calculations, which are based on gossip.

This algorithm is performed in several rounds (bottom-up traverse of the horizontal search tree), and both phases are applied to each shot. In the first phase of the k^{th} round, the nodes obtain the local support counts for a *k-sized* itemset.

Then each node, in each cluster, sends its generated itemsets with their corresponding support values to the head cluster, and each head cluster calculates and keeps the union of received itemsets with the aggregated value on their supports.

After that, a Gossip-based aggregation phase is applied to the head cluster nodes, and the global frequency is obtained. Then, each node can announce its frequent itemsets over the global frequency.

4.1. Bitwise Horizontal Bottom-Up Methods

Frequent pattern mining algorithms are classified into three types of methods. One of the examples is Apriori-based algorithms that perform data mining by generating candidate itemsets and checking whether they are frequent or not with repetitious scans (Agrawal et al., 1993). The second category includes FP-Growth-based algorithms (Han et al., 2000), which use a structure named FP-Tree. They perform data mining without generating candidate itemsets and repetitious scans. Some instances of Apriori-based and FP-Growth-based algorithms were introduced. The third category includes the algorithms which use bitmap techniques (Sohrabi and Barforoush, 2012; Sohrabi and Barforoush, 2013; Burdick et al., 2001; Song et al., 2008).

A Bitwise method is used in the proposed algorithm. In the horizontal approach, each transaction database includes some rows. Each row of bit matrix corresponds to one of the transactions of the transaction database in data mining processing. Each node generates a bit matrix locally. *L* is designated for rows, and *K* is the column number that includes the items. Each row contains a *K*-bit string.

An International Journal
JSCDSS
E-ISSN: 2289-8603

Tid	itemset
1	a, b, d, e, f
2	a, b, c, d, e
3	d, e
4	a, b, c, d
5	a, c
6	d, e, f

Fig. 1. An instance of the transaction database in each node.

If each row has *i* items, an i^{th} bit of the desired string is considered to be one, otherwise zero. First, an array is constructed with the bit matrix, which is named *Colsum*. It stores the support of each item. Then, the matrix columns and the algorithm orders matrix items in descending order with the help of *Colsum*. Then, we generate the horizontal search tree according to the arranged matrix. An instance of a transaction database in each node is indicated in Fig. 1. First, the bit matrix of this TDB is generated. The initial bit matrix is shown in Fig. 2 without pruning.

	a	b	c	d	e	f
1	1	1	0	1	1	1
2	1	1	1	1	1	0
3	0	0	0	1	1	0
4	1	1	1	1	0	0
5	1	0	1	0	0	0
6	0	0	0	1	1	1
Colsum	4	3	3	5	4	2

Fig. 2. The initial bit matrix without pruning.

The matrix columns are arranged in descending order by *Colsum* values, and then the horizontal search tree is constructed. This concept is indicated in Fig. 3. A set of all items with a different mixture of items are considered. We called our proposed algorithm, which is used by nodes as Local Horizontal Pattern Mining (LHPM).

The generated tree has *K* levels (the number of items). With a depth traverse from level 1 (*1*-itemset) to the node (*K*-itemset), the frequency of itemsets is acquired and then given to the head cluster. The head clusters enter the second phase of the proposed distributed algorithm in *K* rounds. It is explained in the Frequent Distributed Mining section. Then, the global minsup is obtained and given to all local nodes by the head clusters.

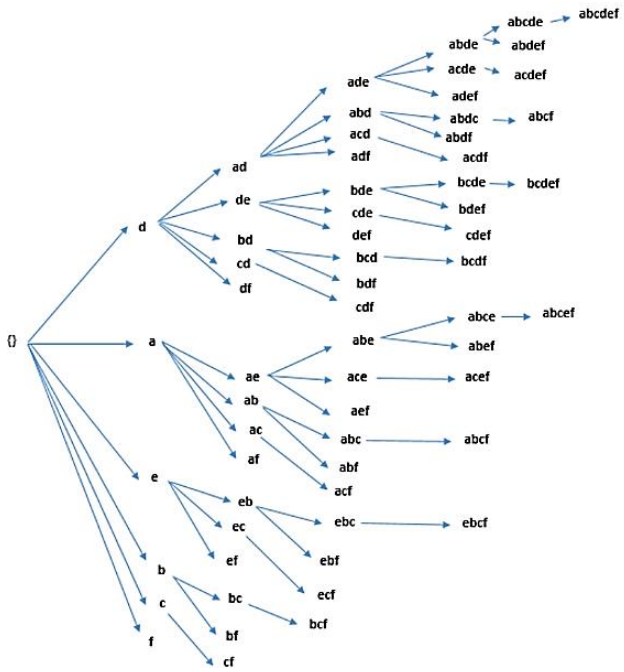


Fig. 3. Horizontal search tree.

LHPM algorithm continues its process after obtaining the global minsup and discovers the frequent itemsets. In the proposed LHPM algorithm and after getting minsup, the horizontal search tree is pruned by the global minsup. It means that a *ColAndVector* is generated for each pattern through a bottom-up model. Therefore, an operator must be applied to all columns of *X* items, and *ColAndVector* is constructed to determine whether the item is sent frequently. If the total values of *ColAndVector* are equal or higher than minsup then we have a frequent itemset; otherwise, it is infrequent. For instance, when *cd* is infrequent, cdh is infrequent, too, according to Apriori regulation. This regulation is used to prune the horizontal search tree in Fig. 4.

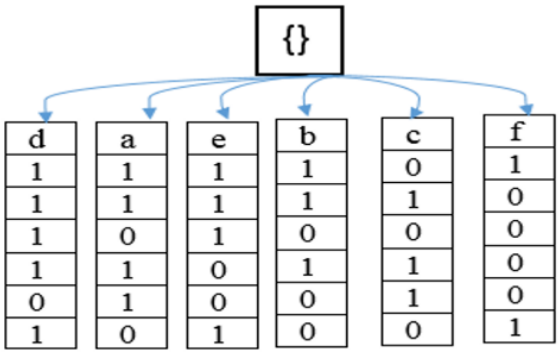


Fig. 4. The first level of the minsup-level in the pruned tree.

In Fig. 5, it is indicated that *cd* is pruned from the tree (*SUP cd* < *3*). The *ade* and all of its children are pruned from the tree. Besides, *bde* and all of its children pruned, too. The pruned tree is indicated in Fig. 6. All the nodes of this tree are the relevant frequent itemsets that are obtained through the LHPM algorithm.

An International Journal
JSCDSS
E-ISSN: 2289-8603

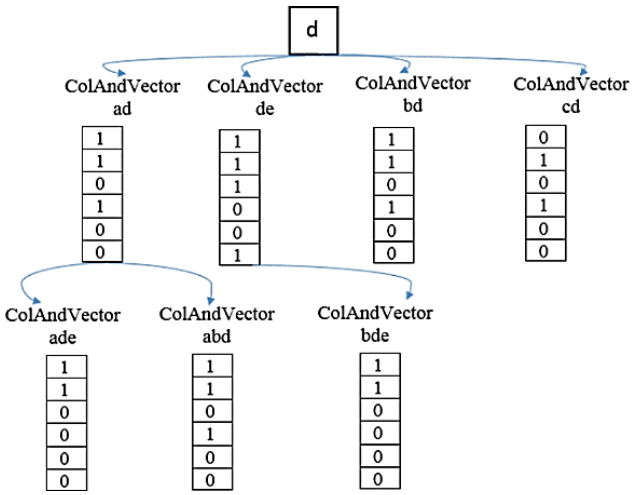


Fig. 5. Horizontal bottom-up mining.

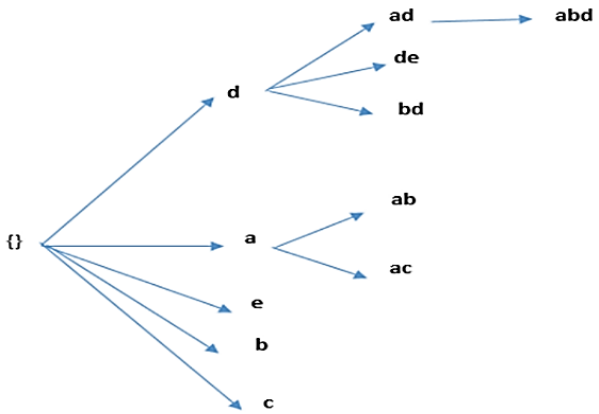


Fig. 6. Pruned horizontal bottom-up tree.

4. 2. Gossip-Based Aggregation

The main objective of aggregating local support counts is to generate global support for each itemset (Bagheri, 2012). At the end of this phase, all nodes know about the global support of itemsets like the ultimate value. They can generate itemsets with the desired length. At the end of the gossip phase, the nodes have enough information to prune inappropriate sets in their local database, according to the value of determined min-sup. Throughout the gossip stage, head clusters exchange information repeatedly with peers who are selected. Up to the time that head clusters achieve near value to the support value of itemset, the procedure is continued. The global support for the itemset *x* in the node *i* is indicated with $V_i(x)$. Before starting the gossip phase of $SUP_i(x) = V_i(x)$, each node receives its local support from its head cluster, and $V_i(x)$ approaches $SUP(x)$ in the gossip procedure. In other words, because of the aggregation of all local nodes by running a couple of rounds of gossip procedure, the cluster attains the global value. In every single series, the correlation of peer head clusters like *i* and *j* of *k*-membered itemsets of these two nodes are compared. If there is a common itemset such as *x* in these lists, the head cluster node will send the support values of that item, which are $V_i(x)$ and $V_j(x)$ to the other head cluster. Therefore, it updates its support value by receiving it from another node. In other words, data transferred among the nodes are *k*-membered itemsets, accompanied by their support values, in the round *k*.

Since the selection of head clusters is entirely random in the gossip phase, and it is possible to reselect a node, a variable named weight is used. The parameter of weight or $W_i(x)$ will remove the effect of reselections and cause the values of the global supports of itemsets to converge on an integer value. This variable is kept in each cluster per each item summation. It means that we will have an array that has the same size as the itemsets array for the parameter of weight.

In each gossip round, each head cluster, such as *i*, performs the updating operations by receiving the support values from other head clusters as Eq. (3):

$$V_i(X)=\left\{V_i(X)+\sum_j V_i(X)\right\}*\frac{1}{2}$$

$$W_i(X)=\left\{W_i(X)+\sum_j W_i(X)\right\}*\frac{1}{2}$$

At the end of the gossip phase, $V_i(X)/W_i(X)$ will be an approximation for the support value of *X* in the node *i* $(Sup(x))$. When the head clusters being aware of this value and giving this value to the sub-nodes of its head cluster, each node can prune its itemsets. It means that if the condition $Sup(x)<min\text{-}sup$ is maintained for one itemset, this itemset is not frequent so that it will be pruned; otherwise, it will be considered as a frequent itemset. This operation and the execution of the algorithm continue until the generated frequent itemsets are not empty.

Since the gossip phase requires more time to converge all nodes on the global response, it is applied only to the head clusters. Using the LEACH algorithm to the nodes in the network and selecting the head cluster, which is responsible for the communication of its cluster to the other clusters existing in the system, has a significant impact on the algorithm efficiency.

5. Experimental Results

We developed a multi-thread java application to implement our algorithm. In our program, every single thread is a symbol of a node, and our buffer is hard disk to let the nodes to communicate. We set up the structure to simulate the time of data transfer and delays. To simulate the structure, we used Mac book pro with Intel core i7 processor, and 8 GB ram and the maximum thread number is 2500.

We used Accident, Kosarak, and Mushroom dataset by Roberto Bayardo from the UCI datasets, and the details of the three datasets are indicated in Table 1.

An International Journal
JSCDSS
E-ISSN: 2289-8603

Table 1

UCI datasets.

UCI datasets			
Datasets	size	Transaction#	Item#
Mushroom	0.56M	8124	119
Accident	16.3M	49046	2113
Kosarak	30.5M	990002	41270

We consider the communication time between nodes in the gossip phase: the rate for data transmission is 100 Mbps and time of transferring data for a typical message size $|M|$ is $|M|/ (100000000/8)$ second.

In our implementation, we assume that all the data is distributed generally among the nodes and measured the convergence time, which is a time dedicated to detecting all frequent patterns. As the number of nodes is increased, it causes a decrease in local processing time. In contrast, more computation time is required for the gossiping phase. Fig. 8 shows the result for Kosarak, Mushroom, and Accident datasets that prove the scalability of our algorithm.

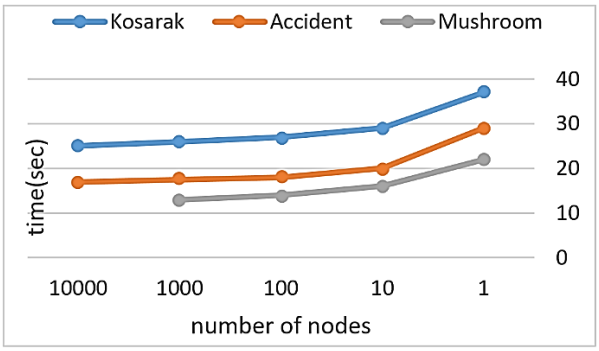


Fig. 8. Execution time comparison for Kosarak, Mushroom, and Accident datasets. Min-sup=0.95.

Fig. 9 compares the execution time of our algorithm before and after using LEACH clustering. The number of nodes is varied from 1 to 10,000. As it is shown in Fig. 9, applying the clustering reduces the time needed to execute the algorithm because this design reduces the number of nodes that contribute to the gossiping phase, when the number of nodes increased, the most difference between the execution time before clustering and after clustering.

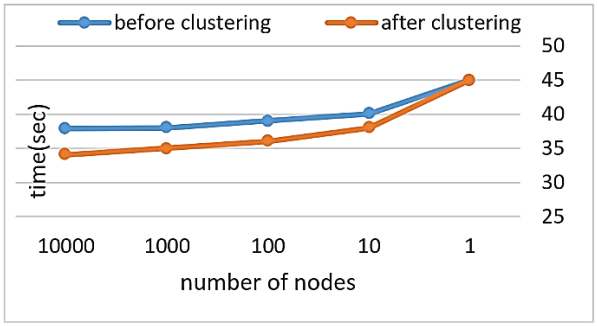


Fig. 9. Execution time comparison for Kosarak dataset, before, and after clustering.

Fig. 10 shows the average accuracy, and the accuracy measures the number of correctly identified frequent itemsets. Our algorithm achieved high accuracy in initial rounds. It can extract all frequent itemsets with more rounds, so our algorithm should execute for more round to discover all frequent patterns of larger sizes.

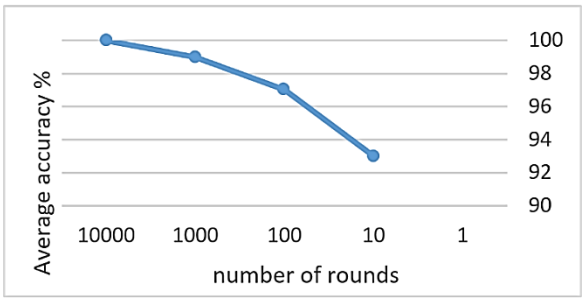


Fig. 10. Average accuracy against the number of rounds, Kosarak dataset, number of nodes=1000.

Fig. 11 compares execution time in our algorithm with a Gossip-based algorithm (Bagheri, 2012). It is a distributed, Apriori based algorithm and uses three layers of data structure to find frequent itemsets. The main difference between it and our algorithm is performing the Bitwise method as the local frequent itemset mining mechanism in each node that resulted in a significant reduction in local processing time in our proposed algorithm. Our experimental results show that our designed system is scalable in terms of the network size and node number.

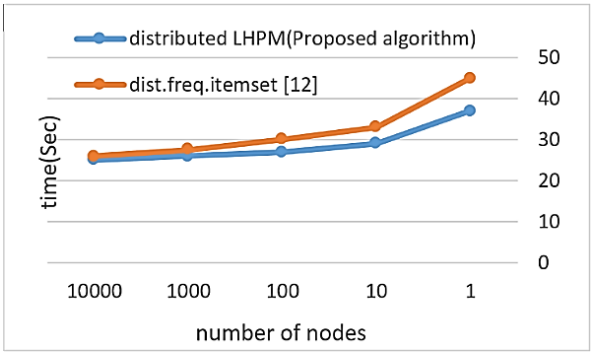


Fig. 11. Comparison of execution time for the Kosarak dataset.

6. Conclusion and Future works

In this paper, we proposed a new, distributed, frequent itemset mining algorithm using the Bitwise approach in the wireless sensor network. Wireless sensor network nodes build clusters and apply the LEACH algorithm. In this way, head clusters choose Gossip-based protocol to link each other. As a result, each node locally success to extract frequent itemsets at higher speed and accuracy. Results from implementation considerably show execution time decreasing compared to another distributed Gossip-based

An International Journal
JSCDSS
E-ISSN: 2289-8603

algorithm. Then, we analyze the average accuracy and execution time in different datasets. For the future, as further study, we consider developing a frequent pattern mining process of the proposed algorithm for dynamic datasets in wireless sensor networks.

Reference

Agrawal, R., & Shafer, J. (1996). Parallel Mining of Association Rules. IEEE Trans. Knowledge and Data Eng, 962-969.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data (pp. 207-216). IEEE.

Ashrafi, M., Taniar, D., & Smith, K. (2004). ODAM: An optimized distributed association rule mining algorithm. IEEE distributed systems online.

Bagheri, M. M.-H. (2012). Mining Distributed Frequent Itemsets Using a Gossip-Based Protocol. In 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing (pp. 780-785). IEEE.

Burdick, D., Calimlim, M., & Gehrke, J. (2001). Mafia: A maximal frequent itemset algorithm for transactional databases. In Proceedings 17th international conference on data engineering (pp. 443-452). IEEE.

Cheung, D., Han, J., Ng, V., Fu, A., & Fu, Y. (1996). A fast distributed algorithm for mining association rules. Fourth International Conference on Parallel and Distributed Information Systems (pp. 31-42). IEEE.

Guan, H., & Ip, H. (2007). A study of parallel data mining in a peer-to-peer network. Concurrent Engineering, 281-289.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. ACM sigmod record, 1-12.

Heinzelman, W., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. 33rd annual Hawaii international conference on system sciences (p. 10). IEEE.

Izadi, V., Shahri, P., & Ahani, H. (2020). A compressed-sensing-based compressor for ECG. Biomedical engineering letters, 1-9.

Lin, C., Li, W., Chen, J., Chung, W., Chung, S., & Lin, K. (2019). A Distributed Algorithm for Fast Mining Frequent Patterns in Limited and Varying Network Bandwidth Environments. Applied Sciences, 1859.

Lin, K., & Chung, S. (2015). A fast and resource-efficient mining algorithm for discovering frequent patterns in distributed computing environments. Future generation computer systems, 49-58.

Lin, K., & Lo, Y. (2013). Efficient algorithms for frequent pattern mining in many-task computing environments. Knowledge-Based Systems, 10-21.

Mohamed, A., Modir, A., Shah, K., & Tansel, I. (2019). Control of the Building Parameters of Additively Manufactured Polymer Parts for More Effective Implementation of Structural Health Monitoring (SHM) Methods. Structural Health Monitoring.

Mohamed, A., Modir, A., Tansel, I., & Uragun, B. (2019). Detection of Compressive Forces Applied to Tubes and Estimation of Their Locations with the Surface Response to Excitation (SuRE) Method. 9th International Conference on Recent Advances in Space Technologies (RAST) (pp. 83-88). IEEE.

Otey, M., Wang, C., Parthasarathy, S., Veloso, A., & Meira, W. (2003). Mining frequent itemsets in distributed and dynamic databases. Third IEEE International Conference on Data Mining (pp. 617-620). IEEE.

Park, B., & Kargupta, H. (2002). Distributed data mining: Algorithms, systems, and applications.

Schuster, A., Wolff, R., & Trock, D. (2005). A high-performance distributed algorithm for mining association rules. Knowledge and Information Systems, 458-475.

Sohrabi, M., & Barforoush, A. (2012). Efficient colossal pattern mining in high dimensional datasets. Knowledge-Based Systems, 41-52.

Sohrabi, M., & Barforoush, A. (2013). Parallel frequent itemset mining using systolic arrays. Knowledge-Based Systems, 462-471.

Song, W., Yang, B., & Xu, Z. (2008). Index-BitTableFI: An improved algorithm for mining frequent itemsets. Knowledge-Based Systems, 507-513.

Sumalatha, S., & Subramanyam, R. (2020). Distributed mining of high utility time-interval sequential patterns using the MapReduce approach. Expert Systems with Applications.

Surakanti, S., Khoshnevis, S., Ahani, H., & Izadi, V. (2019). Efficient Recovery of Structural Health Monitoring Signal based on Kronecker Compressive Sensing. International Journal of Applied Engineering Research, 14(23), 4256-4261.

Tanbeer, S., Ahmed, C., & Jeong, B. (2009). Parallel and distributed frequent pattern mining in large databases. 11th IEEE International Conference on High-Performance Computing and Communications (pp. 407-414). IEEE.

Wolff, R., & Schuster, A. (2004). Association rule mining in peer-to-peer systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2426-2438.

Zaïane, O., El-Hajj, M., & Lu, P. (2001). Fast parallel association rule mining without candidacy generation. IEEE international conference on data mining (pp. 665-668). IEEE.

An International Journal
JSCDSS
E-ISSN: 2289-8603