Computing and Informatics, Vol. 34, 2015, 458–472

DISTRIBUTED FREQUENT ITEM SETS MINING OVER P2P NETWORKS

Zahra Farzanyar, Mohammadreza Kangavari

Department of Computer Engineering
Iran University Science and Technology (IUST)
Tehran, Iran
e-mail: {Z_farzanyar, kangavari}@iust.ac.ir

Abstract. Data intensive peer-to-peer (P2P) networks are becoming increasingly popular in applications like social networking, file sharing networks, etc. Data mining in such P2P environments is the new generation of advanced P2P applications. Unfortunately, most of the existing data mining algorithms do not fit well in such environments since they require data that can be accessed in its entirety. It also is not easy due to the requirements of online transactional data streams. In this paper, we have developed a local algorithm for tracing frequent item sets over a P2P network. The performance of the proposed algorithm is comparatively tested and analyzed through a series of experiments.

Keywords: Data stream mining, frequent item set mining, any time algorithm, distributed data mining, P2P data mining

1 INTRODUCTION

There are many systems where data is distributed over a large and dynamic network of nodes including no special server or client nodes. An interesting example for large scale distributed systems is peer-to-peer (P2P) systems which generate huge mass of data every few seconds.

Popular P2P systems such as Gnutella, e-Mule, Napster, and Freenet can share vast amounts of information. For example, media files such as songs and videos are shared through Gnutella. Participants in the network reveal the files they store on their computers to the system and gain access to files shared by their peers in return [1].

Early consumers in file sharing P2P networks focused on the value of the data while more recent research argues that the consumers will notably profit from the knowledge locked in the data [1]. For instance, it may be discovered that people who download the movie Titanic also look for songs by Celine Dion. Such knowledge can then be discovered in different ways, much like the well-known data mining example stating that "customers who purchase diapers also buy beer". In this case, [2] also refers to an interesting application case from media-file-sharing P2P networks knowing the frequent song set shared or most frequently searched songs helps to distinguish the most popular style of music shared or searched (and hence, recommendations), or what kinds of songs are shared or downloaded together. Thus, the new generation of advanced P2P applications demands support for advanced data mining [2].

We consider mining of frequent item sets that has been the focus of great interest among data mining researchers.

Standard data mining algorithms [3] are not useable in P2P networks unless the total data is centralized in one place. A scenario which centralizes the total data is not scalable because any change must be announced to the central peer. Also the data changes at a faster ratio than the rate at which it can be centralized. However, that data is distributed so widely that it will usually not be feasible to collect it for central processing. It must be processed in place by distributed algorithms suitable to this kind of computing environment [1].

There is large body of literature for frequent item set mining in distributed environment. However, most of these works deal mining in a small scale distributed environment.

Data mining in P2P networks needs different kinds of algorithms because the P2P environment introduces some new problems. The first problem is that there can be no global synchronization [1]. The second problem is that global communication is impossible in large P2P systems. Therefore, the nodes (by node we mean peer) should communicate through a local negotiation. Another problem comes from the dynamic nature of large scale systems in which a node departs or joins the system in mid-computation. A further complication comes from the dynamic nature of data stream. A data stream is an unbounded sequence of data elements continuously generated at a rapid rate and have a data distribution that often changes with time [4].

In this paper, we develop an algorithm which coordinates frequent item sets, mined locally at each node, with the neighbor nodes to discover at each node, global frequent item sets. The proposed algorithm can manage changing data stream and network. The goal of our algorithm is to converge rapidly toward the exact results. To the best of the authors' knowledge this is one of the first attempts on developing a local algorithm to mine frequent item sets in P2P systems whose nodes receives a continuous sequence of transactions.

The rest of the paper is organized as follows: Section 2 briefly describes the related work in mining frequent item sets in P2P networks. Section 3 introduces the problem definition. In Section 4, the proposed algorithm is presented. Section 5 describes the experimental results. Finally, Section 6 concludes the paper.

2 RELATED WORKS

P2P data mining is a new research area. Computing for P2P networks can be classified into three main fields:

1. best effort heuristics,
2. gossip based computations,
3. local algorithms [5].

Recently, local algorithms have been developed for several data mining tasks: association rule mining [1], L2 thresholding [6], outliers' detection [7], decision tree induction [8], meta-classification [9], k-means clustering [10] and multivariate regression [11].

A related work to this paper, Wolff et al. [1] address the problem of association rule mining, bypassing the frequent item set mining problem based on 'majority voting' protocol. In their approach, each node executes a majority voting protocol with its neighbours to determine whether an item set that exists in local data is frequent globally. If a pair of nodes does not agree on majority of an item set, they exchange the local support and local data size. The protocol terminates when there is no message in the total network, and every node then knows whether an item set that exists in its own data is frequent or not. They do not address the problem of frequent item set mining directly, and discover association rules directly on the basis of local communication with neighbouring nodes. Their algorithm for discovering association rules from a P2P network can take communications of the order of network size for those rules whose confidence is about equal to the confidence threshold.

The problem of finding frequent item sets in a P2P network is already addressed in [2] using a sampling-based probabilistic approach. The main idea is to identify the most frequent item sets using a uniform random sample drawn from the whole data. One particular randomly selected node, NS, leads the execution of the P2P-AFIMI algorithm. They assume other node in the P2P network is reachable from NS. NS first collects a uniform sample of data to create a set of item sets that are possibly frequent in the data distributed in the P2P network. This set is called a 'candidate set'. Then it runs standard Apriori algorithm [12] on the sample to generate item sets that are frequent in the sample. To identify item sets actually frequent (globally frequent) in the entire network from the candidate set, node NS takes another round of node sampling to collect local support for each candidate item set and corresponding local data size and estimates the global frequency of each candidate set. The communication between NS and any sampled node happens directly in peer-to-peer fashion. The size of the node sample is determined by desired accuracy level. They assumed that the network topology and data do not change during the execution of the algorithm. This assumption makes their algorithm inappropriate for real P2P networks. Thus using Apriori algorithm causes a decline in the efficiency in this algorithm.

None of the presented algorithms consider P2P network which peers get a stream of tuples.

3 PROBLEM DEFINITION

The problem of on-line mining of recent frequent item sets in data streams is defined as follows: Let $D = T_1, T_2, \dots, T_m$ be a sequence of transactions, where each transaction $T_i \in D, i \in [1, m]$, is a subset of I. $I = \{i_1, i_2, \dots, i_n\}$ is a set of n distinct literals, called items which are units of information in an application domain. An item set x is a subset of items. A window W consists of the most recent N transactions that slides forward for every transaction. The support of an item set x over W, denoted as $\mathrm{Support}(x, W)$, is the number of transactions in W which contain the item set x. An item set x is called a frequent item set if $\mathrm{Support}(x, W) \geq \mathrm{minSupp} \times N$, where minSupp is a user defined minimum support threshold in the range of $[0, 1]$.

In P2P global frequent item set mining, the dataset, in essence, is partitioned among a set of nodes. Each node to mine recent global frequent item sets in the entire recent data over a P2P network can only be associate with a small set of other nodes – its immediate neighbors. By globally frequent item set, we imply an item set whose frequency over the entire current recent data in the network exceeds minSupp. Since, each node in a P2P network receives a continuous sequence of transactions; it may not be possible to propagate the changes to the entire system at the rate they occur. Thus, it is useful if an incremental algorithm can obtain ad hoc results quickly and enhance them as more data is propagated. Such algorithms are called anytime algorithms [1]. The performance of an anytime algorithm is quantified by its average recall and precision [1, 2]. An anytime algorithm is said to be correct if during static periods, in which the dataset and the system do not change, both the average recall and the average precision converge to one [1]. Here, in this paper, we raise convergence rate by selecting super nodes and decreasing communication load through sending only MFIs and MIFIs.

There are several protocols advised for distributed super node selection in an unstructured P2P network. We refer the H2O protocol [13] to select super nodes and their member nodes from an unstructured P2P network. In this way, after a while, every node is either a super node with a list of its member nodes and a list of other super nodes in the neighbourhood, or is a member node that knows its super node(s). In this case, as shown in Figure 1, the communication topology of the P2P network takes the form of some hub consisting of super nodes directly connected to each other in the neighbourhood. We assume the selected super nodes do not change with time. This protocol is avoided in the pseudo code in order to keep conciseness.

In this work, we assume that an underlying mechanism maintains a communication tree that spans all super nodes in a P2P network, communication among peers is reliable and that a super node is informed of changes in the status of adjacent nodes. We further assume that each peer has the same set of attributes for its data tuples. These assumptions are not uncommon and have been made elsewhere in the distributed algorithms literature [1].

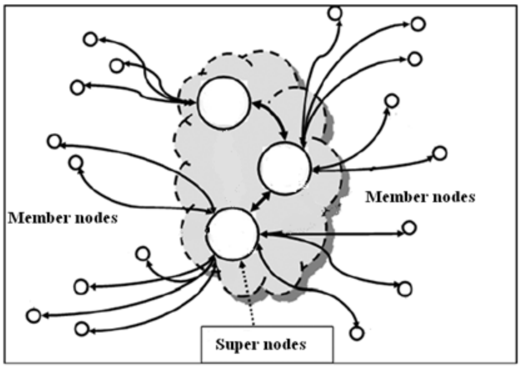

Figure 1. The communication topology

4 THE PROPOSED ALGORITHM

In [1] it has been shown that a P2P frequent item set mining algorithm can be considered as a decision problem in which the participating nodes must decide whether or not each item set is frequent. However, the algorithm presented in that work is only suitable for P2P systems whose nodes can examine their data element several times and face with very little change in data. We present here FIM-P2P algorithm which works well for P2P systems in which each node receives a continuous sequence of transactions, thus the nodes cannot access the entire local data, and each data element can be examined at most once. The purpose of FIM-P2P algorithm is to make certain that each node converges toward the correct result. The algorithm dictates how nodes react when the data concept changes, a member node is separate or joined.

In our algorithm each member node independently mines its local data stream and sends the current result to its super node. Sending of all produced frequent item sets to super nodes results in high communications load. In order to decrease the volume of communications, in our algorithm, each member node only sends Maximal Frequent Item sets (MFIs). The infrequent item sets in each peer may have the chance to become frequent on the recent data of the whole network. In order to prevent any false negative errors it is necessary that each member node sends also information related to infrequent ones. For this purpose, in our algorithm, each member node also sends a part of infrequent item sets, Minimal InFrequent Item sets (MIFIs).

Definition 1. Maximal Frequent Item sets are the sets of item sets that are frequent, but any of their immediate supersets are not frequent. They are denoted by MFIs [4].

Definition 2. Minimal InFrequent Item sets are the sets of item sets that are not deemed frequent, but all their immediate subsets are frequent. They are denoted by MIFIs [4].

Note that in these definitions, maximal and minimal refer to the size of the item set. MFIs whose support count is close to minSupp may cause false negative errors in network, because they do not have the exact support count of their subsets. If a great change occurs in the concept of data and the behaviour of these MFIs change compared to previous stage, this may lead to false negative errors in network about their subsets. In order to decrease such errors, the member nodes can send a few of the subsets of the described MFIs. Consequently, more frequent item sets are sent to the super node, leading to more communication load and less error. How to select these subsets is among our future work.

Therefore, each member node produces MFIs and MIFIs from its current window periodically by using Max-FISM algorithm [4] and in the event of their change as compared to previous stage; it sends them along with their support count to its super node. We assume the size of window in all peers to be the same.

Max-FISM (Maximal-Frequent Item sets Mining) algorithm uses a prefix tree-based summary data structure called Max-Set for maintaining the number of independent appearance of each transaction in the current window. Finally, the set of recent maximal frequent item sets and recent minimal infrequent item sets are obtained from the current Max-Set. The Max-FISM algorithm is avoided in the pseudo code in order to keep conciseness.

Each super node uses a data structure called global tree, denoted as T_G^u based on Max-Set [4] to keep current information of the nodes. Whenever the super node u receives new information, denoted as $\Delta_{new}^m < \text{MFIs} - \text{MIFIs list} >$, from the member node m, it inserts it in T_G^u and deletes the previous information, $\Delta_{old}^m < \text{MFIs} - \text{MIFIs list} >$. Also when a member node is separated from super node, the related information is deleted from T_G^u. Therefore, the algorithm responds to every change in network topology or data.

Each super node for finding global frequent item sets should consider not only the local data, but also data in the other nodes in network. For this purpose, the global tree of super node u must be in agreement with the global tree in each neighbour super node that makes it certain that all nodes that are reachable from another one converge toward the correct result according to their combined dataset. If two global trees are in agreement with each other about their borderline item sets, both of them produce the same mining result. The borderline item sets are MFIs and MIFIs associated with the global tree. In this case, whenever a super node faces the request of a user, it can deliver global frequent item sets by examining its global tree.

The super node u in the node related to item set x in T_G^u keeps the current support count of item set x and network size N_{Px} that are computed by using following functions. N_{Px} is recalculated through changing N_m (number of member nodes) and/or changing N_{Ps} (number of network nodes which each neighbor super node s knows). We symbolize the local support (the support obtained from the member nodes) of global tree as $\mathrm{supp}_{xu}^{\perp}$. The initial support of item set is local support.

$$\begin{aligned}\mathrm{Supp}_{initial} &= supp_{x\,u}^{\perp}\\ N_{Px\,initial} &= N_m\end{aligned}$$

The super node u, to coordinate its global tree with its neighbours, periodically examines its global tree by using Max-FISM algorithm [4] to produce the current MFIs and the current MIFIs and their support count. We show these item sets are denoted Δ^u. Then super node u, for any item set x available at Δ^u, considers Δ_x^{us}. Δ_x^{us} shows the behaviour of item set x, which super nodes u and s has recently reported to each other. The super node u records, for every neighbor super node s, the last message it sent to s about item set x, $< x$, $supp_x^{us}$, ${N_P^{us}}_x >$ and the last message it received from s, $< x$, $supp_x^{su}$, ${N_P^{su}}_x >$ in a buffer. Super node u computes the following function of these messages:

$$\Delta_x^{us} = \Delta_x^{su} = \langle (supp_x^{us} + supp_x^{su}), ({N_P^{us}}_x + {N_P^{su}}_x)\rangle \tag{1}$$

Δ_x^{us} is recalculated when a message is sent to or received from s. Super node u coordinates its global tree with super node s for item set x by keeping the same value Δ_x^{us} (except for the time a message travels between u and s, $\Delta_x^{us} = \Delta_x^{su}$). During the time that ($\Delta_{y\supseteq x}^{us} \geq minSupp$, where y is x or superset of x, and $\Delta_x^u \geq minSupp$) or ($\Delta_{y\subseteq x}^{us} <$ minSupp, where y is x or subset of x, and $\Delta_x^u <$ minSupp) there is no need for u and s to interchange data about item set x. It means that super nodes s and u are convergence on item set x. If, on the other hand, $\Delta_{y\subseteq x}^{us} < \mathrm{minSupp} \leq \Delta_x^u$ or $\Delta_x^u < \mathrm{minSupp} \leq \Delta_{y\supseteq x}^{us}$ or $\Delta_x^{us} = \Delta_x^{su} = 0$, super node u should send message $< \mathrm{supp}_x^u - \mathrm{supp}_x^{su}, N_{Px}^u - {N_P^{su}}_x >$ to super node s. The super node s receiving new information deletes the previous information of the super node u, in T_G^u and adds new information as shown in the functions 4 and 5. After this message is sent, $\Delta_x^u = \Delta_x^{us}$. When no message is sent, s knows that u is in agreement with it on item set x.

$$\begin{aligned}\mathrm{Supp}_x &= \mathrm{Supp}_x - {\mathrm{supp}_x^{us}}_{old} + {\mathrm{supp}_x^{us}}_{new}\\ N_{Px} &= N_{Px} - N_{Px\,old}^{us} + N_{Px\,new}^{us}\end{aligned}$$

For example, in the global tree of node u, AB is a MIFI where min Supp = 2/5, as shown in the Figure 2 a). We assume $N = 1$ and $N_m = 5$. The neighbor node s finds AB as a frequent item set, then it sends message $(AB, \mathrm{Supp} = 4, N_P = 5)$

to u. Then the global tree of node u is changed based on Max-Set tree [4] as shown in the Figure 2 b).

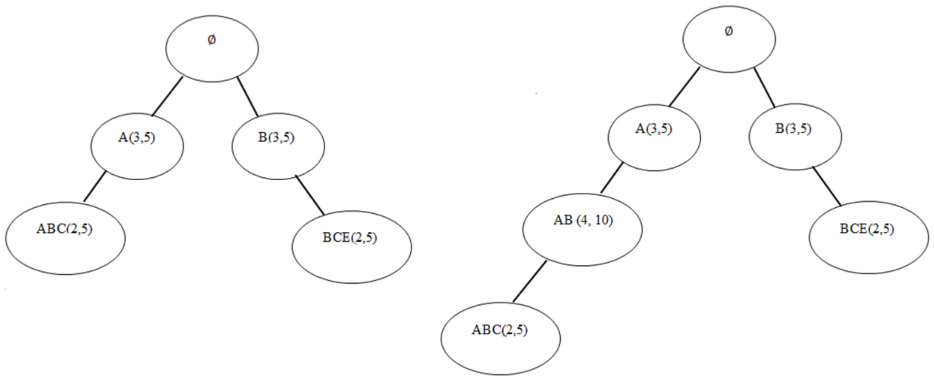

Figure 2. A running example

Each super node executes the algorithm independently with each of its immediate neighbor super nodes. Consequently, super nodes will become aware about current status of item sets in more number of network nodes. When no message is sent across network, nodes have been convergence toward the right result about each item set x, i.e., that upon convergence $\Delta_x^u \geq minSupp$ if and only if the support count obtained of the nodes which can be reached from u (as shown in functions 6 and 7) is not less than minSupp. This proof is given in the following. We show the group of edges connected to u as E^u including the edges of neighbor super nodes.

$$\begin{aligned} \mathrm{Supp}_x &= \mathrm{supp}_x^{\perp u} + \sum_{su \in E^u \text{ if } s \text{ and } u \text{are not in agreement about } x} \mathrm{support}_x^{su} \\ N_{Px} &= N_m + \sum_{su \in E^u \text{ if } s \text{ and } u \text{ are not in agreement about } x} N_{Px}^{su} \end{aligned}$$

Assume a connected tree T(V,E) with data $\Delta_x^u =< \mathrm{supp}_x, N_{Px} >$ about item set x at super node $u \in V$. Assume Δ_x^{us} the value after the algorithm has converged. For each $u \in V$ let $[u] = \{s \in V : s$ be reachable from u using edges in $E\}$. For every $us \in E$ let $[u]_s = \{w \in V : w$ be reachable from s using edges in $E - \{us\}\}$. Finally, for any subset of super nodes $S \subseteq V$ let $\Delta_x^S =< \mathrm{supp}_x^S = \sum_{s \in S} \mathrm{supp}_x^{\perp s}, N_P^S = \sum_{s \in S} N_m^s >$.

Theorem 1. In a tree $T(V,E)$ upon convergence, for all $u \in V$, $\Delta_x^u \geq$ minSupp if and only if $\Delta_x^V \geq$ minSupp.

Lemma 1. In a tree T(V,E) upon convergence if for some u, $\Delta_x^u \geq$ minSupp then for each s such that $us \in E$, $\frac{\mathrm{supp}_x^{su}}{N_{P\,x}^{su}} \approx \Delta_x^{[u]_s}$, where $\approx$ means agreement.

Proof. By induction on $|[u]_s|.|[u]_s| = 1$. It means that $[u]_s = s$ and s received message only from than u and sent message only to u. So, $\Delta_x^{[u]_s} =< \text{supp}_x^{\perp_s}, N_m^s >$ and $\Delta_x^s =< \text{supp}_x^{us} + \text{supp}_x^{\perp_s}, N_P^{us}{}_x + N_m^s >$. We assume by contradiction $\text{supp}_x^{su}/N_P^{su}{}_x \asymp \Delta_x^{[u]_s}$, where $\asymp$ means disagreement. It inferences that $\Delta_x^{us} = \Delta_x^{su} = (\text{supp}_x^{us} + \text{supp}_x^{su})/(N_P^{su}{}_x + N_P^{us}{}_x) \asymp (\text{supp}_x^{\perp_s} + \text{supp}_x^{us})/(N_{Px}^s + N_m^s) = \Delta_x^s$. Therefore, super node s should send a message to u. It is contradictive to the convergence assumption. Assuming the Lemma 1 is true for $[u]_s \leq K$ we prove that it is true for $[u]_s = K+1$. $\Delta_x^s = (\text{supp}_x^{\perp_s} + \sum_{ws \in E^s} supp_x^{ws} \ /N_m^s + \sum_{ws \in E^s} N_{p_x}^{ws}) = (\text{supp}_x^{\perp_s} + \text{supp}_x^{us} + \sum_{ws \neq us} \text{Supp}_x^{ws})/(N_{P_x}^{us} + N_m^s + \sum_{ws \neq us} N_{p_x}^{ws})$. Because of the contradictive assumption, $\text{supp}_x^{su}/N_{P_x}^{us} \asymp (\text{supp}_x^{\perp_s} + \sum_{ws \neq us} \text{Supp}_x^{ws})/(N_m^s + \sum_{ws \neq us} N_{P_x}^{ws})$, that follows $\Delta_x^{us} = \Delta_x^{su} = (\text{supp}_x^{su} + \text{supp}_x^{us})/(N_{P_x}^{us} + N_{P_x}^{su}) \asymp ((\text{supp}_x^{\perp_s} + \text{supp}_x^{us} + \sum_{ws \neq us} \text{Supp}_x^{ws})/(N_{P_x}^{us} + N_m^s + \sum_{ws \neq us} N_{P_x}^{ws})) = \Delta_x^s$. Again, super node s should send a message to u. It is contradictive to the convergence assumption.

Proof of Theorem 1: We know that if super node u has $\Delta_x^u \geq$ minSupp and no super node $s \in V$ needs to send a message, then every super node s must have $\Delta_x^u \geq$ minSupp. We now must prove that $\Delta_x^u \geq$ minSupp if and only if $\Delta_x^V \geq$ minSupp. We assume an imaginary super node h with $\text{supp}_x^{\perp_h} = N_m^h = 0$ to the super node u. Note that if u does not need to send a message to h then $\Delta_x^h = \Delta_x^{uh} = \Delta_x^u$ and also that h never sends messages. Upon convergence, according to the Lemma 1, $\text{minSupp} \leq \Delta_x^h = \Delta_x^{uh} \approx \Delta_x^{[h]_u} = \Delta_x^V$. Since we know $\Delta_x^h \geq \text{minSupp}, \Delta_x^V$ is also greater than or equal to minSupp. □

Defining the significance degree of item set x as $\text{sig} = (\sum_{u \in V} \text{supp}_x^{\perp_u} / \sum_{u \in V} N_m^u)$ $-$minSupp, we will show in Section 5 that in our algorithm even a minor significance is enough to get a correct result about an item set using data from only a small number of super nodes. Therefore, our algorithm has a good locality. Also, during the execution of the algorithm, each node keeps an ad hoc result. During static periods of network, most of the nodes will converge toward the correct results rapidly. These features make it scalable and therefore suitable for P2P networks where peers get a stream of tuples rapidly. The pseudo code of the FIM-P2P is given in the following.

Algorithm: (FIM-P2P)
Input for Super node u: (1) The set of super nodes that is located in a certain geographical distance with it E^u and (2) A user-defined minimum support threshold minSupp $\in [0,1]$.
Ad hoc output of Super node u: Current global tree in the super node u: T_G /* The algorithm never terminates*/
Initialization: $T_G \leftarrow a$ global tree with null initialization; $N_m^u = 0$; /* The number of member nodes of u */
Main: Repeat the following forever
On receiving a message $\Delta^m <$ MFIs – MIFIs list $>$ from member node m: /* the current MFIs and MIFIs in member node m*/
If there is $\Delta_{old}^m <$ MFIs – MIFIs list $>$ into the buffer then Remove MFIs –

```
MIFIs list$\Delta^m_{old}$ from global tree;
Else $N_m = N_m + 1$; /* the member node $m$ is joined to super node $u$ */
For each $x \in$ MFIs − MIFIs list
If $x \notin T_G$ Create a new entry of form $(x, \mathrm{supp}_{x_{\text{MFIs−MIFIs list}}}, N_m)$ into the $T_G$;
Else $\mathrm{supp}_x = \mathrm{supp}_x + \mathrm{supp}_{x_{\text{MFIs−MIFIs list}}}$ ;
Record $\Delta^m <$ MFIs − MIFIs list $>$ as last message of $m$ into the buffer;
On receiving a message $< x, \mathrm{supp}^{su}_x, N^{su}_{P_x} >$ from super node $s$: /* information is about item set $x$ */
If $x \notin T_G$ Create a new entry of form $(x, \mathrm{supp}^{su}_x, N^{su}_{P_x})$ into the $T_G$;
Else If $< x, \mathrm{supp}^{su}_x, N^{su}_{P_x} >_{old}$ is into last recorded messages into buffer;
$\mathrm{supp}_x = \mathrm{supp}_x - \mathrm{supp}^{su}_{x_{old}}$; $N_{P_x} = N_{P_x} - N^{su}_{P_{x_{old}}}$ ;
$\mathrm{supp}_x = \mathrm{supp}_x - \mathrm{supp}^{su}_x$; $N_{P_x} = N_{P_x} + N^{su}_{P_x}$; Record $< x, \mathrm{supp}^{su}_x, N^{su}_{P_x} >$ into the buffer;
On failure of member $m$:
Remove MFIs − MIFIs list$_{\Delta^m_{old}}$ from global tree;
Remove MFIs − MIFIs list$_{\Delta^m_{old}}$ from last recorded messages;
$N_m = N_m - 1$; /* the member node $m$ is disjoined from super node $u$ */
Periodically: /* Find Global MFIs − MIFIs list($T_G$) by Max-FISM algorithm:
For each $x \in$ MFIs − MIFIs list
For each super node $s \in E^u$
$\Delta^{su}_x = \delta^{us}_x = (\mathrm{supp}^{us}_x + \mathrm{supp}^{su}_x)/(N^{us}_{P_x} + N^{su}_{P_x})$ /* the real support count of $x$, which $u$ and $s$ has recently reported to each other */
```

5 EXPERIMENTAL RESULTS

For testing effectiveness of FIM-P2P, we implemented a simulated network including 2 000 peers connected in a random tree spread over on a 50×40 grid. The 200 super nodes are selected from the network by using H2O protocol [13] and do not change with time in these experiments. In these simulations all processors have the same speed and all messages are delivered in unit time. These assumptions are not because the algorithm requires such limitations but because properties such as convergence and locality are best demonstrated in these conditions [1]. For lack of real datasets of the scale required by a system of 2 000 computers, we used synthetic databases generated by IBM [12]. We generated synthetic database, T30.I20.1000K, where the parameters of synthetic data denote the average transaction size (T), the average maximal frequent item set size (I), and the total number of transactions (D), respectively. We split this data homogeneously between member nodes. We assume the size of window (W) in all member nodes to be 300 transactions.

The two main qualities measured in our experiments are the convergence rate of the result and the communication cost. The convergence rate is determined by calculating the recall and precision in the ad hoc results of algorithm and that is measured by a centralized algorithm, given all the recent data of all of the peers. We have measured the communication cost of the algorithm based on the produced communication volume of average peer to converge to the exact results. Since P2P network scale is very large, any algorithm which does not have good average locality cannot be regarded scalable. Therefore, a locality is used to evaluate performance [1]. In Experiment 1, our aim is to consider locality of the proposed algorithm.

To evaluate the performance of FIM-P2P, we compare it with the Majority-Rule algorithm [1]. Since, Majority-Rule algorithm does not address the problem of frequent item set mining, and discover association rules directly. To create an analogous environment to compare the two algorithms, we have changed Majority-Rule algorithm to discover frequent item sets directly. In these experiments, minSupp is set at 0.3.

Experiment 1. In one static state, we consider the environment of super node u for item set x as the super nodes whose information about item set x is gotten by super node u. The locality of an algorithm is measured according to the worst-case size and the average size of the environments of the different super nodes, because super nodes just must be coordinated together in the network. The average environment size determines the number of messages exchanged between super nodes. Figure 3 indicates locality of algorithm. For each algorithm, the worst state and average state of the environment size are shown against item set significance.

The environment size of each super node for an item set depends on the significance degree of item set. While the significance of an item set is close to zero, in order to achieve a precise global result, more super nodes must be coordinated together in the network which environment size is large. As shown in the Figure 3, when item set significance moves away from zero, the size of the environment decreases. In Majority-Rule algorithm, when significance degree of an item set is about 0.16, i.e., its global support degree is larger than minSupp by 0.16, the algorithm enjoys a desirable locality. The minimal size of environment is about 220 nodes and the average size of environment is about 80 nodes. In Majority-Rule Algorithm, when the significance degree of an item set is zero, the environment size, in the worst state, equals to the size of network, because all nodes must be coordinated together in the network. But in FIM-P2P Algorithm, all super nodes must be coordinated together in the network. The number of super nodes is smaller than the number of nodes in the network. This is regarded as a major weakness of Majority-Rule algorithm for marginal item sets. In FIM-P2P Algorithm, when significance degree of an item set is about 0.05, the algorithm enjoys more desirable locality. The minimal size of environment is about 95 nodes and the average size of environment is about 30 nodes. According to results obtained, in our algorithm, dependency of locality on significance would be decreased to a great extent compared with Majority-Rule algorithm and even in significance degree close to zero has a good local behaviour. This decrease in the size of environment of each node for each item set proves better locality of our algorithm. A good locality would cause that algorithm is fast and communication efficient [1].

Further to locality, two main characteristics of the proposed algorithm are namely convergence rate and communication cost.

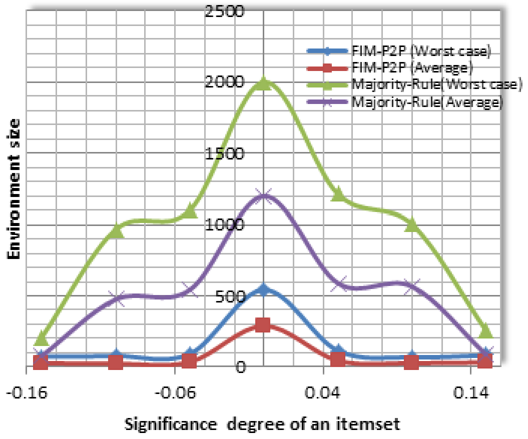

Figure 3. Locality of FIM-P2P algorithm with increasing environment size

Experiment 2. We calculate convergence rate by calculating the recall and precision in the ad hoc results. To evaluate the correctness of the outcome of FIM-P2P, we compare it with the frequent item sets generated by Max-FISM algorithm run on the entire recent data. Figure 4 compares the convergence of the recall a) and of the precision b) in FIM-P2P and Majority-Rule with increase of environment size in three significant degree – 0.15, 0.05 and zero.

In both algorithms, the frequent item sets with high significance degree, it is expected that it may be generated earlier and nodes agree upon the same quickly. This is also correct about the infrequent item sets with very low significance degree. Because they are usually formed due to noise, as more nodes enter the calculations, they come down rapidly [1]. In Majority-Rule algorithm, item sets whose support count is very close to minSupp are generated only when the algorithm works for a long period. If a huge number of item sets are the same as described earlier, execution time of algorithm will be very long. This defect with the Majority-Rule algorithm makes it incompatible with peer-to-peer networks whose peers receive the continuous sequence of transactions. In our algorithm, as shown in Figure 4, convergence speed of nodes toward correct results is higher compared to Majority-Rule algorithm. This has been caused due to use of super nodes and each super node communicates with its neighbours only about its MFIs and its MIFIs instead of all its candidate item sets. These results show that in the proposed algorithm the convergence rate of peers toward correct results is higher compared to Majority-Rule algorithm. High convergence speed is one of the essential requirements of peer-to-peer networks where network topology and data constantly change.

Experiment 3. Figure 5 compares the entire communication load generated in each super node in order to achieve the entire global results by FIM-P2P with

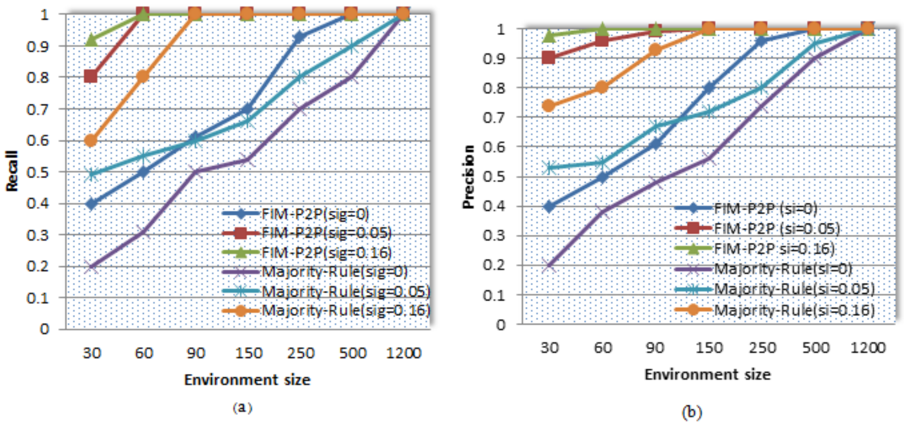

Figure 4. Precision and Recall with increasing environment size

communicating load of each node in Majority-Rule algorithm through increase of size of network. As shown in Figure 5, in FIM-P2P algorithm in each size of network, the entire generated communication load in each node is less that of Majority-Rule. Since Majority-Rule algorithm uses Apriori property [12] for generating the candidates, communication is established among nodes for all candidates. Moreover, when a great change occurs in the concept of data, the behaviour of many candidates will change compared to previous stage. This may lead to re-communication load on system. However, in our algorithm, all nodes release only their MFIs and MIFIs and speed of release of the said information is higher because of use of super nodes. This causes that less messages would be generated. Also as shown in the Figure 5, upon increase of size of network, the communication load generated in FIM-P2P algorithm shows less growth compared with Majority-Rule algorithm. This achievement is caused due to use of super nodes. Consequently, the results obtained from this experiment confirm that our algorithm enjoy higher scalability with respect to the number of nodes.

6 CONCLUSION

We presented a local algorithm, FIM-P2P, for discovering P2P network-wide frequent item sets. We have shown that our algorithm has a good locality which causes fast convergence of the result and low communication load. Such algorithms are needed for the next generation P2P applications such as P2P file sharing networks, P2P web mining, P2P bioinformatics and P2P astronomy using national virtual observatories.

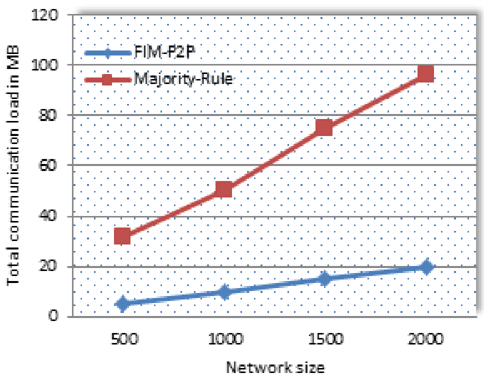

Figure 5. Total communication in megabytes with increasing P2P network size

REFERENCES

[1] Wolff, R.—Schuster, A.: Association Rule Mining in Peer-to-Peer Systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 2004, pp. 2426–2438.

[2] Datta, S.—Kargupta, H.: A Communication Efficient Probabilistic Algorithm for Mining Frequent Item Sets from a Peer-to-Peer Network. Statistical Analysis and Data Mining, 2009, pp. 48–69.

[3] Farzanyar, Z.—Kangavari, M.: Efficient Mining of Fuzzy Association Rules from the Pre-Processed Dataset. Computing and Informatics, Vol. 31, 2012, No. 2, pp. 331–347.

[4] Farzanyar, Z.—Kangavari, M.—Cercone, N.: Max-FISM: Mining (Recently) Maximal Frequent Itemsets over Data Streams Using the Sliding Window Model. Computers and Mathematics with Applications, Vol. 64, 2012, No. 6, pp. 1706–1718.

[5] Datta, S.—Bhaduri, K.—Giannella, C.—Wolff, R.—Kargupta, H.: Distributed Data Mining in Peer-to-Peer Networks. IEEE Internet Computing, Vol. 10, 2006, No. 4, pp. 18–26.

[6] Wolff, R.—Bhaduri, K.—Kargupta, H.: Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems. Proceedings of SDM '06, Bethesda, MD, 2006, pp. 430–441.

[7] Branch, J.—Szymanski, B.—Gionnella, C.—Wolff, R.—Kargupta, H.: In-Network Outlier Detection in Wireless Sensor Networks. Proceedings of ICDCS '06, Lisbon, Portugal, July 2006.

[8] Bhaduri, K.—Wolff, R.—Giannella, Ch.—Kargupta, H.: Distributed Decision Tree Induction in Peer-to-Peer Systems. Proceedings of Journal Statistical Analysis and Data Mining, Vol. 1, 2008, No. 2.

[9] Luo, P.—Xiong, H.—Lu, K.—Shi, Z.: Distributed Classification in Peer-to-Peer Networks. Proceedings of KDD '07, San Jose, California, USA, 2007, pp. 968–976.

[10] Datta, S.—Giannella, C.—Kargupta, H.: K-Means Clustering over a Large, Dynamic Network. Proceedings of the SIAM Conference on Data Mining (SDM), Bethesda, MD, 2006, pp. 153–164.

[11] Bhaduri, K.—Kargupta, H.: An Efficient Local Algorithm for Distributed Multivariate Regression in Peer-to-Peer Networks. Proceedings of the SIAM Conference on Data Mining (SDM), Atlanta, GA, 2008, pp. 153–164.

[12] Agrawal, R.—Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), 1994, pp. 487–499.

[13] Lo, V.—Zhou, D.—Liu, Y.—Gauthier Dickey, C.—Li, J.: Scalable Supernode Selection in Peer-to-Peer Overlay Networks. Second International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P 2005), La Jolla, CA, 2005, pp. 18–25.

Zahra Farzanyar received the Ph.D. degree in software engineer from the Iran University of Science and Technology, Tehran, Iran, under the supervision of Dr. Mohammadreza Kangavari. Her Ph.D. thesis was on Developing Frequent Itemsets Mining Algorithms in Large scale Peer to Peer Environments with Data Stream Theory. Her research interests include data stream mining and distributed and peer to peer data mining.

Mohammadreza Kangavari received the B.Sc. degree in mathematics and computer science form the Sharif University of Technology in 1982, the M.Sc. degree in computer science from Salford University in 1989, and the Ph.D. degree in computer science from the University of Manchester in 1994. He is currently a lecturer in the Computer Engineering Department, Iran University of Science and Technology, Tehran, Iran. His research interests include data mining, machine learning, and natural language processing.