Egyptian Informatics Journal (2015) **16**, 297–307

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

FULL-LENGTH ARTICLE

A partition enhanced mining algorithm for distributed association rule mining systems

A.O. Ogunde [a,*], O. Folorunso [b], A.S. Sodiya [b]

[a] *Department of Computer Science, Redeemer's University, Redemption Camp, Ogun State, Nigeria*
[b] *Department of Computer Science, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria*

Received 17 September 2014; revised 29 May 2015; accepted 30 June 2015
Available online 11 September 2015

KEYWORDS

Data mining;
Distributed association rule mining systems;
Frequent itemset;
Database partitioning;
Mobile agents

Abstract The extraction of patterns and rules from large distributed databases through existing Distributed Association Rule Mining (DARM) systems is still faced with enormous challenges such as high response times, high communication costs and inability to adapt to the constantly changing databases. In this work, a Partition Enhanced Mining Algorithm (PEMA) is presented to address these problems. In PEMA, the Association Rule Mining Coordinating Agent receives a request and decides the appropriate data sites, partitioning strategy and mining agents to use. The mining process is divided into two stages. In the first stage, the data agents horizontally segment the databases with small average transaction length into relatively smaller partitions based on the number of available sites and the available memory. On the other hand, databases with relatively large average transaction length were vertically partitioned. After this, Mobile Agent-Based Association Rule Mining-Agents, which are the mining agents, carry out the discovery of the local frequent itemsets. At the second stage, the local frequent itemsets were incrementally integrated by the from one data site to another to get the global frequent itemsets. This reduced the response time and communication cost in the system. Results from experiments conducted on real datasets showed that the average response time of PEMA showed an improvement over existing algorithms. Similarly, PEMA incurred lower communication costs with average size of messages exchanged lower when compared with benchmark DARM systems. This result showed that PEMA could be efficiently deployed for efficient discovery of valuable knowledge in distributed databases.

1. Introduction

Data mining is the process of extracting hidden and useful information and rules from large databases [1]. Many businesses and organizations now depend on such patterns and rules in order to make crucial decisions [2]. Association rule mining (ARM) is a very significant data mining technique. It tries to find frequent associations, correlations, patterns or

[*] Corresponding author.
E-mail address: ogundea@run.edu.ng (A.O. Ogunde).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2015.06.006

casual structure sets of items or objects form such databases. Much attention has been given to association rule miming by the data mining community. An example of this association is indicated in how the purchase of a group of items, called an itemset, affects the purchase of another group of items [3]. In today's world, most databases are now of distributed nature where each local database stores its ever increasing amount of day-to-day data. In view of this, the use of centralized data mining methods to discover useful patterns in such organizations' data may not be realistic again because merging datasets from different sites into a centralized site leads to huge network communication costs. Distributed data mining has thus emerged as an active subarea of data mining research. There is therefore an urgent need for distributed data mining systems that would reduce response times and communication costs of distributed association rule mining systems. Mobile agents are relatively autonomous entities capable of carrying out actions in an environment perceived by it. They are threads of control that have the capability to trigger the transfer of arbitrary code to any remote computer. Therefore, in this paper a mobile agent-based distributed association rule mining with a partitioned enhanced mining algorithm was proposed.

2. Data Mining (DM)

Extracting useful and understandable information from data in databases is the main concern of data mining [4]. Machine learning algorithms may be deployed to search for patterns that may be exhibited in the data and subsequently deriving the knowledge. Data mining techniques include association rules mining, clustering, classification, sequential pattern mining, time series mining, etc. According to Gharib et al. [5], Association Rule Mining (ARM) has received significant attention in the research community. These ARM methods had shown very interesting results when applied in many problem areas. According to Ahangar et al. [6], the main components of data mining such as artificial intelligence, statistics and machine learning have developed for many years and the development of these techniques and technology has made data mining very practical and effective for discovery of valuable knowledge in data storage environments. A number of DM algorithms had been proposed in this area but none can be said to be the best in all situations [7].

2.1. Distributed Association Rule Mining (DARM)

DARM is the semi-automatic pattern, knowledge and rules extraction from distributed data sources [8]. The exponential growth of data stored in organization's databases does not make it feasible for all data to be resident in the memory as obtainable in most existing DARM systems. Also, it has become a matter of necessity for data to be inherently distributed for fault tolerance purposes. In a multi-database mining environment, often one needs to handle multiple large databases. Solutions to such problems could be obtained through the analysis of local patterns [3].

2.2. The data partitioning approach to association rule mining

Savasere et al. [9] were the first to propose a Partition algorithm for ARM but this algorithm does not have a distributed nature and it's not agent based. In this context, a partition of the database is any subset of the transactions contained in that database. Other definitions of local and global partitions are similar to those defined in the study by Savasere et al. [9]. Other researchers like Albashiri et al. [10], Coenen and Leng [11], and Pudi [12] also worked on partitioning. Algorithms proposed by most of these researchers only tried to reduce the number of false candidate items as quickly as possible but they did not consider giving priority to the partitioning of the original datasets as a major factor in improving the mining process. Nguyen and Orlowska [13] suggested that a closer look at the data could be a way of improving the overall performance of such systems. In this work, an improved approach to the data partitioning method is proposed based on the combination of both horizontal partitioning and vertical partitioning of the dataset. This would greatly reduce the number of Local frequent itemsets, thereby leading to a higher common Global candidate for the partitions as partition sizes were calculated to fit into the available memory spaces. Researchers that have used similar methods had not really shown the basic factors considered for partitioning the DB and other factors considered in the division of the data to smaller partitions. In addition, many of these algorithms are deployed directly on remote, stand-alone machines and not in a distributed setting and are therefore not agent-based in nature. This is also another major area of contribution in this paper as the proposed algorithm is basically agent-based and is also suited for a distributed setting.

2.3. Data partitioning and distribution in ARM algorithms

Though many algorithms had been proposed in the literature for ARM, the big sizes of the databases makes it difficult for the mining task to take place in a single process. This makes it imperative for researchers to come up with more efficient methods for carrying out such mining tasks. It is believed that distributing the large data into smaller partitions will enhance and improve the mining process. Both horizontal partitioning and vertical partitioning of the database exist in the literature. Albashiri [7], Coenen and Leng [11] had worked on the Horizontal Data Distribution and Partitioning Approach. The Vertical Data Distribution and Partitioning Approach was also explored by [7] and [14]. Three strategies were identified by many researchers in the literature [15–17] for DARM algorithms that used the Apriori algorithm or its variants. These methods are count distribution, data distribution and candidate distribution. DARM Algorithms based on data distribution partitions both the candidate itemsets and the database while also using the memory of parallel systems [18]. Parallel/Distributed ARM Algorithms based on candidate distribution partitions candidate itemsets but selectively replicates the database transactions for the independent mining of each of the processes. DARM algorithms based on Count Distribution use a data-parallel method with horizontal partitioning of the database for local scanning and detection of all candidate itemsets [11,19,20]. Usually, partitioning is made possible in many contexts with the use of mobile agents, which are computer software located in a particular environment with the capability of carrying out autonomous actions in this environment as a way of meeting their set objectives [21]. JADE is one of the most important agent-based platforms [1].

3. Methodology

The main reason for multiple scanning of the database by the traditional *Apriori* is because the number of possible itemsets to be tested for support is exponentially large most especially if this is to be done in a single scan of the database. The methodology proposed in this work was based on the terminologies of some earlier work done by Ogunde et al. [22]. It partitioned the entire database into smaller partitions that can be tested and mined in a single scan. This had very great potentials, most especially when the average length of transactions in the database was very short e.g. less than or equal to twenty. PEMA used horizontal segmentation to partition the dataset of databases with lesser transaction lengths. In a single scan, PEMA generated a set of all potentially frequent itemsets by scanning the database through the mining agent and establishing links of the frequent itemsets in a tree structure. This frequent set was a superset of all frequent itemsets, which were frequent itemsets in the local partition and might not be frequent when the whole DB was finally considered. It is also very important to note that no false negatives were reported at that stage. After this stage, integration of the frequent itemsets was carried out by the results integration coordinating agent, itemsets were set up and their actual support was measured without rescanning the database.

3.1. Definitions and notations used in algorithm design

The following definitions were used for the purpose of this work.

A *partition* $p \subseteq$ DB of the database refers to any subset of the transactions contained in the database DB. Any two different partitions are non-overlapping, i.e., $p_i \bigcap p_j = \phi$.

A *local support* for an itemset is the portion of transactions in the partition containing that itemset.

A *local candidate itemset* is an itemset within a partition, which is being tested for the minimum support.

A *local frequent itemset* is an itemset in a partition whose local support is at least the user-defined minimum support. A locally frequent itemset may or may not be globally frequent.

A *global support* for an itemset is the fragment of transactions containing that itemset in a DB.

A *global candidate itemset* is an itemset within the entire DB that is being tested for the minimum support.

A *global frequent itemset* is an itemset whose global support in the entire DB is at least the user defined minimum support.

In order to find all global frequent itemsets, the following notations were used:

C_k^p – Set of local candidate k-itemsets in partition p.
F_k^p – Set of local frequent k-itemsets in partition p.
F^p – Set of all local frequent itemsets in partition p.
C_k^G – Set of global candidate k-itemsets.
C^G – Set of all global candidate itemsets.
F_k^G – Set of global frequent k-itemsets.
F^G – Set of global frequent itemsets.

Individual itemsets were represented by small letters and sets of itemsets are represented by capital letters. Note that the partition number was omitted when referring to a local itemset. The following notations were used: c[l].c[2].……… ……c[k] to represent a k-itemset c consisting of items c[1],c [2], ……c[k].

3.2. The algorithm–PEMA

The structure of the PEMA is described here in this section.

Algorithm: PEMA

```
$P$ = partitioned_database($T$); $n$ = number_of_partitions;
$s$ = minimum_support
// 1st Section
for $i = 1$ to $n$ do {
    read_in_partition($P_i$ in $T$)
$F^i$ = generate all frequent itemsets of $P_i$ using apriori in main
memory}
// Merge Section
for ($k = 2$; $F_k^i \ldots F_k^i \neq \phi$, $i = 1, 2, \ldots, n$; $k++$) do {
$C_K^G := \bigcup_{i=1}^n F_k^i$
          }
// 2nd Section
$F^G = \phi$;
for $i = 1$ to $n$ do {
read in partition($P_i$ in $T$)
∀ candidates $C \in C^G$ {
        if $s\{C\}_{Ti} \geqslant \sigma$ then //check if support of global candidate
itemsets from all partitions is ⩾min-sup
        $F^G = F^G \cup \{C\}$; // generate frequent global itemsets with
$\{C\}$'s that meets the condition above
}
Return $F^G$
//The working procedure for PEMA is presented as follows:
    ($p$: database_partition, $s$: min_sup)
$F_1^p$ = {frequent 1-itemsets with their tidlists}
//tidlists – transaction identifier lists i.e. unique number for
identifying each transaction
for ($k = 2$;$F_k^p \neq \phi$; $k++$){
  ∀ itemsets $f_1 \in F_{k-1}^p${
    ∀ itemsets $f_2 \in F_{k-1}^p${
            if $f_1[l] = f_1[l] \wedge f_1[2] = f_2[2] \wedge \ldots \wedge f_1[k-2] = f_2[k-2]$
$\wedge f_1[k-1] = f_2[k-1]$ then
              $c = f_1[l] . f_1[2] \ldots f_1[k-1] . f_2[k-1]$
          if $c$ cannot be pruned then
            $c$.tidlist = $f_1$.tidlist ∩ $f_2$.tidlist
            if |c.tidlist| / |$p$| ⩾ s then
            $F_{k-1}^p = F_{k-1}^p \cup \{c\}$
          }
    }
}
return $\bigcup_k F_{k-1}^p$
```

3.3. Horizontal partitioning algorithm component of PEMA

There are two phases in the execution of the PEMA. In the first phase of the initial mining task, the mining agent logically divides i.e. horizontally segments the database into a number of non-overlapping partitions, when the database is relatively small as defined in previous sections. This depends solely on the number of available data sites in the system. This is simply done by the horizontal segmentation of the entire data into smaller partitions. Usually, the partitions were mined concurrently by the mining agents at all sites, and all the frequent itemsets for each partition were generated. At the end of the first phase, these frequent itemsets were incrementally merged to generate a set of all potentially frequent itemsets. In the second phase, the actual supports for these itemsets were generated and the frequent itemsets were identified. The partition sizes were chosen in a way that each partition could be accommodated in the main memory. The amount of memory available at each of the data sites as at the time of the first mining also stood as a major consideration in this work for determining how many partitions that could be obtained from the DB. In this work, the mining agent examined the system to obtain the current total available memory space and then used this information to divide the database into the several partitions. This is to ensure that each partition fits into the main memory during the first phase of the mining. Let the entire database is represented by ED, the total available memory space be represented by TAMS, in megabytes. Let the reserved memory be represented by RM (it is proper to reserve a certain percentage of the memory for proper memory usage and management), which is a percentage of the total available memory. If the size of the Mobile Agent-Based Association Rule Mining-Agent (MAARM) equipped with the mining algorithm is represented by MA. In order to calculate the number of partitions, it is appropriate to have all the measurements to the same standard (preferably in megabytes for the purpose of this work). Hence, the number of partitions (NP) from the entire database is given by the formula described here, which is another contribution of this work.

$$\mathrm{NP} = \mathrm{ED}/(\mathrm{TAMS} - (\mathrm{TAMS} * \mathrm{RM}) - \mathrm{MA}) \tag{1}$$

It should be noted that ED, TAMS, RM and MA are not fixed parameters but will usually vary from one site to another. The implication of this is that the value of NP at anytime is dependent on the values of these variables. For instance, if the size of the entire database in a particular data mining site is 20 GB; size of the mining agent is 800 KB; reserved memory is 10%; and the size of the total available memory space is 1800 MB; then

$$\mathrm{NP} = 20000/(1800 - (1800 * 0.1) - 0.8) = 12.35178$$

NP is upwardly approximated to the next whole number. For example, the database in this example will be divided into thirteen partitions.

The next thing is to derive a formula for finding the number of transactions that will be in a particular partition *p*. It was also assumed here that the number of transactions in the entire database is known in advance (this is true for most real life datasets). Therefore, if the number of transactions in the database is represented by NTDB; and the number of partitions as derived above remains NP, then the number of transactions that will be in each partition (NTP) without overlapping is calculated with the formula in Eq. (2).

$$\mathrm{NTP} = \mathrm{NTDB}/\mathrm{NP} \tag{2}$$

For example, if the total number of transactions in the DE for the above example is four hundred and thirty-five thousand five hundred and ninety (435590) transactions and the number of partitions as calculated was 13, then, the number of transactions in each partition will be,

$$\mathrm{NTP} = 435590/13 = 33506.92.$$

This result is downwardly approximated to the nearest whole number, that is, 33506 transactions from the above example. It is obvious in the example that the first twelve partitions will have exactly the same number of partitions while the last partition will have the rest of the transactions which is a little more than the first twelve. This will take care of the effect of the first approximation. This paper assumed that the database is located on secondary storage while the total available memories are also known in advance.

3.4. Vertical partitioning algorithm component of PEMA

To allow the data to be mined using a number of mobile agents requires the allocation of the different data sites to each mining agent by the coordinating agent. In PEMA, for single very large databases with high average transaction length, the data agent vertically partitions the data by dividing the number of columns in the data by the number of available data sites, in order to generate equal number of columns. This has to be done in cases where the data are not already distributed. In such cases, the algorithm performs faster because the time for data partitioning is completely eliminated. In practice therefore and in most cases, it is simplest to divide the data into equal portions. In cases where it is not possible to exactly divide the data into equal columns, the remainder is redistributed from the very first site until no more allocation can be done. That is, if the average transaction length of the data is AVL and we have n data sites available. Thus, we have the number of partitions to be ATL/n. For example, in the case where we have data with twenty-two columns (or average transaction length of twenty-two) to be vertically partitioned among four different data sites. The first data site will have six columns; the second also will have six columns while the third and the fourth will have five columns each. Horizontal partitioning, or segmentation, used in PEMA, is more direct than the vertical partitioning component of PEMA. PEMA assumed that datasets used are uniform and homogeneous; therefore, the number of records was divided by the number of available partitions, which was allocated to each mining agent for mining task. PEMA also optimizes memory usage as earlier on described.

Finally, it should be noted the system proposed in this work works in a dynamic fashion; in that it performs horizontal segmentation or partitioning of the database through the data agent whenever the average length of transactions in the databases is very small. For very large data with long transaction lengths, vertical partitioning of the data was deployed by the data agent to improve the performance of the mining algorithm. This hybrid component of PEMA is a typical representation of the novel method being preached in this work. It combined horizontal segmentation, vertical partitioning and incremental mining of datasets in one method. This was very necessary because real-life databases are usually fragmented in various locations. One major thing unique to this work is that, PEMA could be deployed to mine both real and synthetic datasets already vertically distributed in various data sites and/ or can as well dynamically and vertically partition very large datasets with very long transaction lengths e.g. *Covertype* data form UCI machine learning repository, while distributing them to the various sites depending on the available number of mining sites. This is one of the major tasks performed by the data agent in our architecture. PEMA is very flexible as it does not only perform the global mining task, also has provisions for partial global mining; a situation in which a data miner is not interested in mining all the parts of the data or all the data sites available. This was easily achieved by PEMA as shown in the results presented in section four.

3.5. Message communication in PEMA

Most DARM systems are characterized with a high number of messages exchanged during the mining task. It is therefore important to minimize the number and sizes of messages exchanged in this work. In previous works already described in Section 2, information exchange among agents or processes in the system requires sending of messages between themselves. In our design, an optimized one-to-many data exchange approach was used. Here, agents in the system do not need to exchange data until the final results integration stage. The results integration agent does not have to wait for all mining to complete at the local sites unlike previous works where processes or agents had to wait before they can read their messages and perform their tasks. Each mining agent performed its task locally at the data site and waits for the result integration agent to collate the global results incrementally from one site to another depending on the sites whose local results are available. In PEMA, global results were integrated incrementally from one site to another by the results integration coordinating agent with the total message exchanged representing the number of sites minus one. That is, given n number of sites, the total exchange of data will simply take $n-1$ operations. This is illustrated by Fig. 1. This simply means that given ten data sites, the total number of messages exchanged by the system would be nine. The major benefit of this method is that it involved fewer exchanges of messages. The system even performed better even with increasing number of data sites.

From each partition or data site, PEMA removed infrequent items from the incremental integration of two sites at a time and then left the result in the main memory. During the knowledge integration of the two sites, it inserted the local frequent itemsets into the main memory. While inserting these local frequent itemsets, it checked whether they were already in memory. If yes, it increased the counter for that local frequent itemset by one. Otherwise, it inserted the local frequent itemset into the main memory with a count equal to one. So, given a total number of n sites, the total number of messages exchanged from site to site equals ($n-1$).

$$T_{message\_no} = (n-1)\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

We computed the total message size for the knowledge integration as follows. To compute the total message size (size of the aggregate of sending and receiving sites' messages), take n as the number of sites, F^L as the first local frequent itemsets, F^G as the globally frequent itemsets, $|F^L|$ as the size of locally frequent itemsets (this would be needed for only the first data site prior to the first integration) and $|F^G|$ as the size of globally frequent itemsets (first instance of this was derived on first integration where we had the first intermediate global frequent itemsets). Summation of all the remaining message sizes at the $n-1$ data sites gave us the total message size for the knowledge integration. Thus, we had the formula for the total message size ($T_{message\_size}$) as follows:

$$(T_{message\_size}) = (1 * |F^L|) + \left(\sum_{i=1}^{n-1}(n-i) * |F^G|\right) \quad (4)$$

3.6. DARM scenario with PEMA agents

Agent types used in PEMA are as follows: User Agent (UA), Association Rule Mining Coordinating Agent (ARMCA), Data Agent (DA), Mobile Agent-Based Association Rule Mining-Agent (MAARM), Mobile Agent-Based Result Reporter (MARR), Results Integration Coordinating Agent (RICA), Registration Agent (RA), and Automated Mining Activation Agent (AMAA). All agents were created in the development environment, JADE, and were registered with the registration agent (RA). The User Agent (UA) provided the interface between the system, users and other components of the DARM system. The interface between other components of the DARM system and the input data was provided by the data agent (DA). MAARM agents were processing agents that performed the ARM task at the data sites either automatically or in response to user requests. The DARM process began with either (i) the AMAA automatically initiating the mining process after a specified percentage increment had been achieved in the updated database or (ii) the user initiating the DARM request. This would be taken over by the user agent. In the first case, the AMAA automatically initiated the mining process by starting the ARMCA which started the MAARMs necessary for carrying out the DARM task depending on the number of available data partitions as confirmed by the data agent to the ARMCA. The user agent received notification of user requests and started the ARMCA in the second stage. The ARMCA in turn starts the MAARMs necessary for performing the DARM task depending on the number of available data partitions. ARMCA sends a copy of MAARM to all data sites designated for the mining task. Each MAARM accepts the request, travels to the data partition and begins the mining process in order to generate the local frequent itemsets. Once completed, the MAARM delivered the results to the MARRs, which took only the results information for the local frequent itemsets to the ARMCA

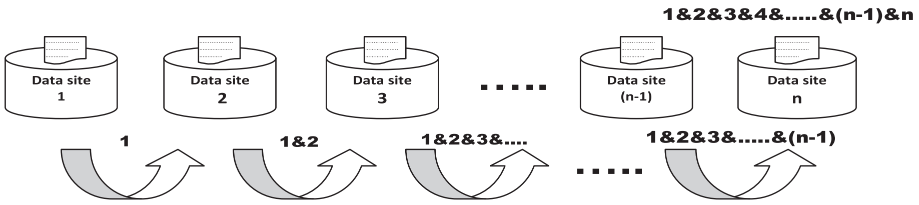

Figure 1 Number of messages exchanged in PEMA.

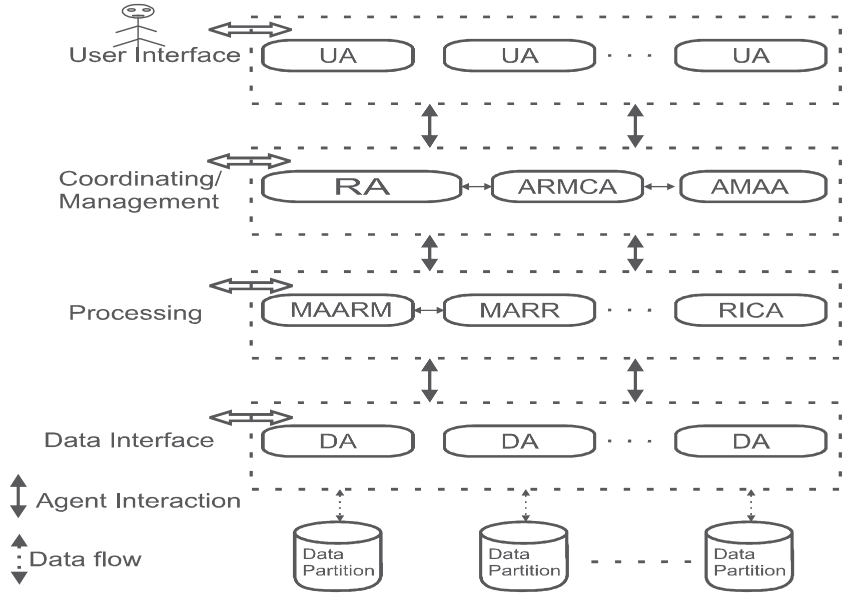

Figure 2 Agent architecture for the system.

for processing and coordination. The ARMCA determined the order of knowledge integration among the results sites using the results information. The outcome of this was sent to the RICA, which performed the knowledge integration at the results sites taking two sites at a time as determined by ARMCA. This was repeated until all the results had been merged together to obtain a global result. The RICA sends the global mining result to the ARMCA which in turn sends it either to the UA or to AMAA depending on which of these two agents that initiated the mining process. This description is summarized by Fig. 2.

4. System implementation

This section describes the simple experiments designed based on the algorithm PEMA described in section three. The input to the system is a DARM task consisting of data situated at either single or multiple data sites. Each individual data server has some specific data and resource requirements, all of which must be satisfied before the mining task can start. Association Rules are to be mined on distributed database servers, each of which had a set of data sizes (possibly different). The set of database servers and the data sizes were static, and this information was available at all servers. However, the load conditions at the servers can vary with time. The aim of the system is to complete all the DARM tasks as fast as possible. Usually, a DARM task is fully executed by the mining agent. A mining agent executing a DARM task migrates to a server that has the data required for the mining task, and tries to generate the frequent itemsets. If all the necessary resources at the data site are available and the environment is conducive, then the mining task is executed. Otherwise, the data server environment is sensed to get an idea of what the environmental change is and the mining agent may have to adapt to the changing environment using the hard-coded instructions programmed into it.

4.1. Description of datasets

Datasets used in this work were real datasets downloaded from the popular UCI Machine Learning Repository hosted by the Centre for Machine Learning and Intelligent Systems [23]. Four major benchmark data popularly used for distributed association rule experiments were downloaded and used for all the experiments. They are Pima-Indian-Diabetes, Letter-Recognition, Connect-4 and Cover Type. Full descriptions of the characteristics of these datasets are shown in Tables 1–4.

Table 1 Characteristics of Pima-Indian-Diabetes data.

Number of items	42
Number of records	768
Average transaction size	8
Area	Life
Date donated	1990-05-09
Number of web hits as at when downloaded	79634
Filename	pima.T768L8

Table 2 Characteristics of Letter-Recognition data.

Number of items	106
Number of records	20,000
Average transaction size	16
Area	Computer
Date donated	1991-01-01
Number of web hits as at when downloaded	77,051
Filename	letRecog. T20000L16

Table 3 Characteristics of connect4 data.

Number of items	130
Number of records	67,557
Average transaction size	42
Area	Game
Date donated	1995-05-04
Number of web hits as at when downloaded	29,935
Filename	connect4. T67557L42

Table 4 Characteristics of covertype data.

Number of items	120
Number of records	581,012
Average transaction size	54
Area	Life
Date donated	1998-08-01
Number of web hits as at when downloaded	39,247
Filename	covertype. T581012L54

4.2. Implementation tools

For the purpose of experimentation with the system developed in this work – PEMA, the following tools were used. Java Development Kit (JDK) 6, Netbeans 7.2 was used for the front-end application. JADE 3.7 was used as the middleware platform for multi-agent system development. VMware Workstation 7.0 virtual machine was used for simulating the distributed database. WampServer version 2.2 with MySQL version 5.5.24 was used for the backend database. Data Discretization/Normalization tool was used for normalizing and discretizing the data for easy mining.

4.3. Implementation of the distributed data sites

A software tool called VMWare Workstation version 7.0.1 was used for creating the distributed environment where the datasets were stored. A total of four data sites were created on three virtual machines and the host system. The configuration of the data sites is displayed in Tables 5–8.

4.4. Data preprocessing

Data discretization and normalization was carried out to preprocess the data for mining. The Data Normalization tool was integrated into the data agent for data discretization and normalization of data used in PEMA. The tool used was an open-source stand-alone Java application by Liverpool University Computer Science-Knowledge Discovery in Data Group (LUCS-KDD) for Data discretization/normalization software Version 2 [24].

Table 5 Description of datasite1 (host system).

Hard disk	400 GB
Memory	3 GB
Processors	2 Core(s)
Guest operating system	Windows 7 Home Premium

Table 6 Description of datasite2.

Hard disk	100 GB
Memory	1 GB
Processors	1
Guest operating system	Windows 7 Ultimate edition

Table 7 Description of datasite3.

Hard disk	100 GB
Memory	1 GB
Processors	1
Guest operating system	Windows 7 ultimate edition

Table 8 Description of datasite4.

Hard disk	100 GB
Memory	1 GB
Processors	1
Guest operating system	Windows 7 Ultimate Edition

4.5. Experimentation

The experiments were designed to analyze the effect of the following: the number of data sources, the size of the datasets in terms of number of records, and the size of the datasets in terms of number of items. All datasets described were used for one experiment or the other. All experiments were performed on four virtual machines running on Intel (R) Core (TM) i5-2450M CPU @ 2.50 GHz, 2501 MHz, 2 Core(s), 3 Logical Processor(s) Pentium(R) with 6 GB of main memory running on Windows 7 Home Premium Edition. Datasets used for the experiment were distributed on the four virtual machines created. The following were measured for one or more of the experiments: (i) response time (seconds/milliseconds), (ii) the communication overhead (number and size of messages exchanged). The experiments were performed by varying the minimum support threshold between the ranges of 0% and 100% of total transactions depending on the particular dataset used. The ARM results of the *PEMA* algorithm described in section three of this work were compared with the performance of other existing state-of-the-art algorithms such as Apriori, AprioriTFP and FP-Growth.

4.6. Results and discussions

The section gives a description of the PEMA system and results obtained from various experiments carried out on the described data. As discussed in Section 3, the system described in this work can perform both Association Rule Mining (ARM) and Distributed Association Rule Mining (DARM). The first part of this section will be dedicated to results obtained by comparing the proposed algorithm PEMA to three already existing algorithms i.e. AprioriT, FP-Growth and AprioriTFP.

4.7. Performance analysis of PEMA algorithms for ARM

Two major data were used for experimenting with the mining algorithms for Local ARM. The data are Pima-Indian-diabetes and letter-recognition data.

4.7.1. Experiments conducted on pima-indian-diabetes data

The first experiment was conducted on pima-Indian-diabetes data, which is a relatively small-sized data with min_sup varied between 10% and 50% of the total transactions while the min_conf was fixed as 80% (Fig. 3). The second experiment was also conducted on same data with min_sup varied between 10% and 50% of the total transactions while the min_conf was fixed as 60% (Fig. 4).

Results obtained from both experiments showed that the response time of PEMA is better than the other three algorithms. It was also noted that the performance was at par with AprioriTFP from 20% min_sup upward. The lesser response times obtained for all algorithms as the min_sup increases was due to the fact that the search space becomes very small as the support increases, which actually accounted for the very low response times.

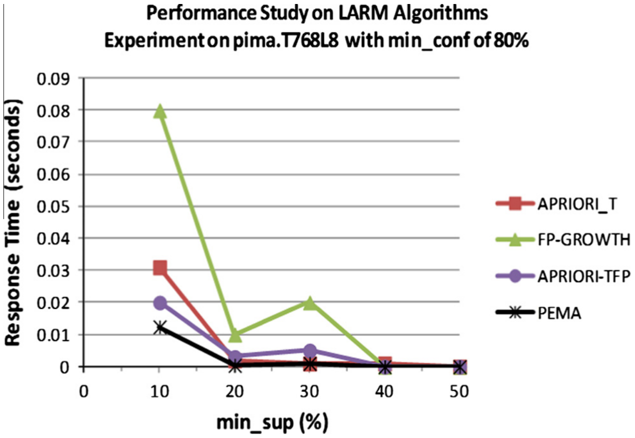

Figure 3 Performance study of algorithms on Pima-Indian-Diabetes data with varying min_sup and 80% min-conf.

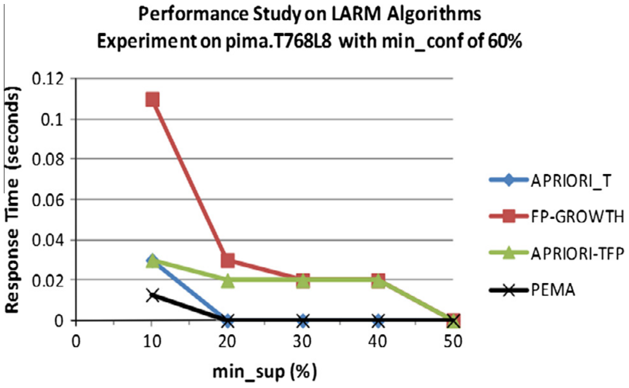

Figure 4 Performance study of algorithms on Pima-Indian-Diabetes data with varying min_sup and 60% min-conf.

4.7.2. Experiments conducted on letter-recognition data

The first experiment was conducted on letter-recognition data, which is a relatively big sized data with about 20000 transactions, with min_sup varied between 10% and 50% of the total transactions while the min_conf was fixed as 80%. The second experiment was also conducted on same data with min_sup varied between 10% and 50% of the total transactions while the min_conf was fixed as 60%. Results shown in Figs. 5 and 6 were obtained. These results showed PEMA performed better than the other three algorithms. It was also clear that there was a gradual convergence between the algorithms as the min_sup increases.

4.9. DARM experiments

DARM experiments were also conducted on the remaining datasets, which are connect4 and covertype. The minimum support of 20% and a minimum confidence of 80% were used as default values for PEMA. The results of these experiments are also shown in the following subsections. It should be noted here that these values could be changed anytime by the user as the need arises.

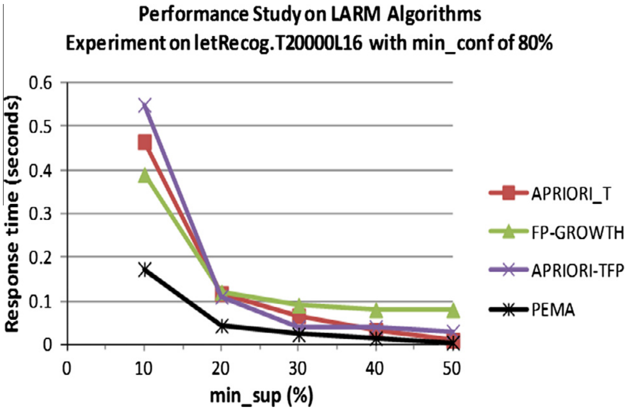

Figure 5 Performance study of algorithms on letter-recognition data with varying min_sup and 80% min-conf.

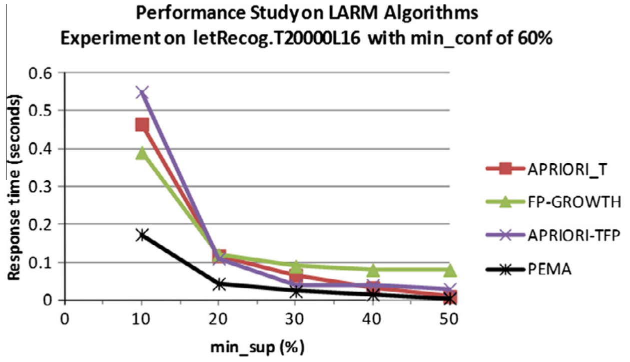

Figure 6 Performance study of algorithms on letter-recognition data with varying min_sup and 60% min-conf.

4.61. Performance study on PEMA with three benchmark algorithms

PEMA was also tested against the results obtained by Venkatesan and Ramaraj [25] on mining connect4 real dataset. The min_sup parameter was varied between 10% and 80% while the min_conf was fixed at 80%. PEMA distributed the connect4 data into four available data sites; carrying out the mining concurrently before obtaining the global result. The results (Fig. 7) showed that the response time of PEMA was best on the data. This was obviously due to the fact that the data have the fewest total number of transactions and also the smallest average number of transactions. Also, connect4 data returned a relatively high response time when the min_sup was low because of the high average transaction length of the data.

4.6.2. Performance study of PEMA on number of sites

Figs. 8 and 9 show the scale-up experiments conducted for PEMA and two other benchmark DARM methods FDM and DDN as presented in the study by Deypir and Sadreddini [20]. The experiments were designed to study how the number of available data sites affects the performance of the algorithms. Connect4 dataset was used in the experiment. The running times and sizes of messages communicated were considered with min_sup and min_conf sets at 80% and 100% respectively. The results of this experiment showed that with the same dataset, the response times of each of the methods improves as the number of distributed sites increases (Fig. 8). This showed that for large datasets, most especially with high average transaction lengths, the algorithm will work better when the data are distributed into a number of sites and mined after which the result is integrated to generate the needed global knowledge. It was also observed that PEMA performed far better than the other two methods; FDM had the worst performance when the number of distributed sites was least i.e. only two sites. The communication overhead in terms of the size of messages exchanged was also measured with regard to same experiment. The results obtained showed that for FDM and DDN, the size of messages exchanged increased with increase in the number of sites (Fig. 9). PEMA also had the best performance with a very low increase in the sizes of messages exchanged as the number of sites increased. This was actually so because PEMA is fully agent based and exchange of messages would only and strictly occur during the knowledge integration stage.

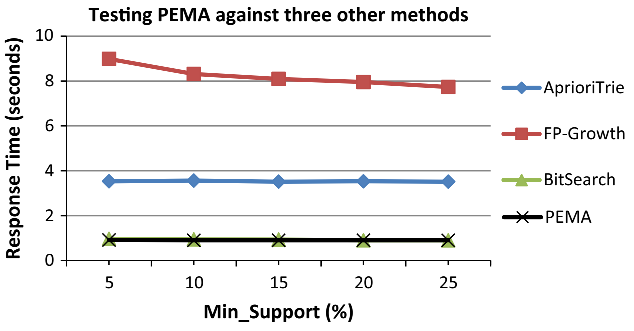

Figure 7 PEMA against three benchmark algorithms.

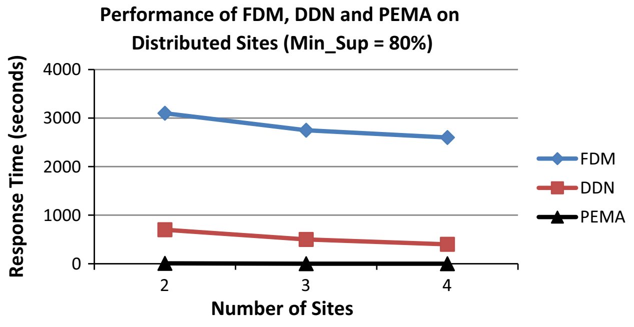

Figure 8 Scale-up experiments on the number of sites (response time).

4.6.3. Performance study based on communicated overhead

Experiments were conducted to measure the communication overhead of the proposed system with two other small real datasets commonly used for same purpose in the literature [26,27]. Fig. 10 shows the result of experiment conducted on Connect4 dataset where the communication overhead in terms of size of messages exchanged was measured with min_sup varied between 75% and 90%, and min_conf set at 100%. The

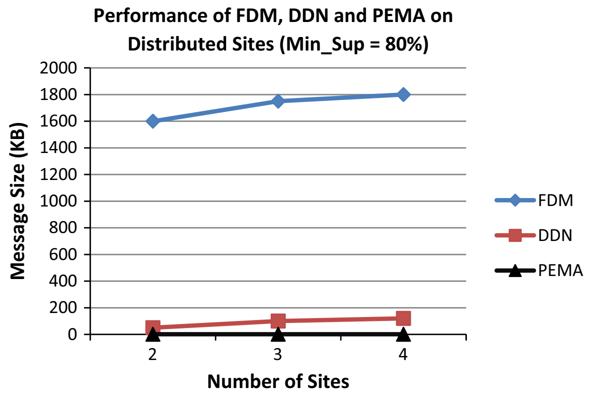

Figure 9 Scale-up experiments on the number of sites (message size).

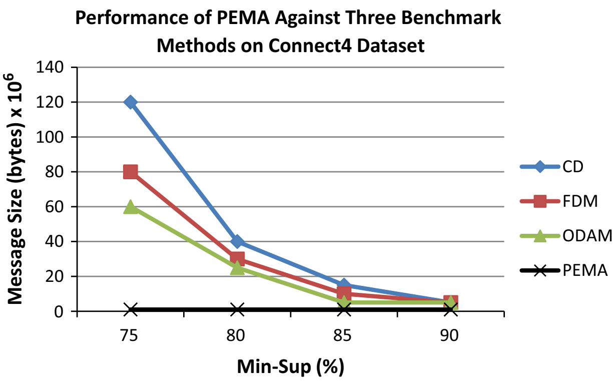

Figure 10 Communication overhead of four DARM methods on connect4 dataset.

results of this experiment showed that with connect4 dataset and 75% min_sup threshold, all the methods exchanged high number of exchanges except PEMA. Also, for all the four methods, the size of messages exchanged improved considerably as the percentage minimum support threshold is gradually increased. PEMA had the best performance even at the 75% min_sup threshold and dropped a little as it increases.

Fig. 11 on the other hand showed the results of similar experiment conducted on the covertype dataset, which was described in Section 4.1. From this result, performance of all the four methods improved with the size of messages exchanged reducing with increase in the minimum support threshold. PEMA had the best performance even at the 75% min_sup threshold with about 1.2 Kilobytes of message exchanged. The size of messages exchanged by PEMA later dropped to about 1 kilobyte with increase in minimum support threshold.

4.6.4. *Performance study of the behavior of PEMA on three different datasets*

Performance of PEMA was also tested using three different real datasets as described earlier on. The min_sup parameter was varied between 10% and 80% while the min_conf was fixed at 100%. Results (Fig. 12) showed that the response time was best on the Pima Indian Diabetes Data. This was obviously due to the fact that the data had the fewest total number of transactions and also the smallest average number of transactions. Also, connect4 data returned a relatively high response time when the min_sup was low because of the high average transaction length of the data.

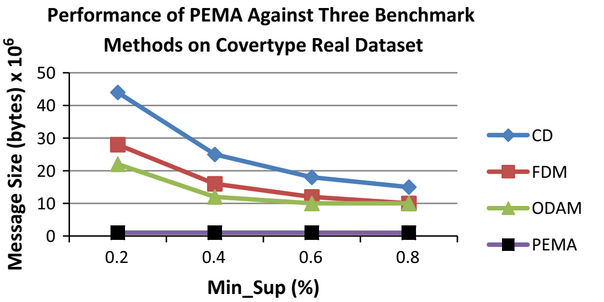

Figure 11 Communication overhead of four methods on covertype dataset.

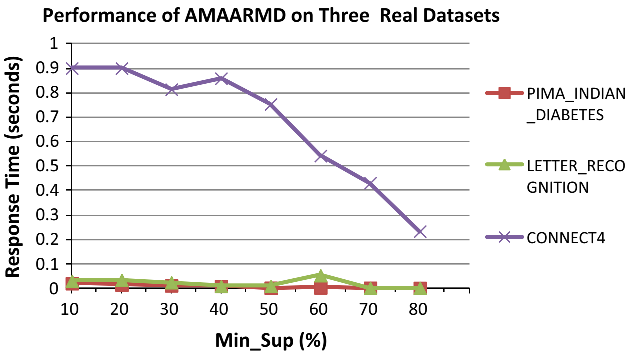

Figure 12 Performance on three real datasets.

5. Conclusion

A study on distributed association rule mining area of the subject data mining was conducted in this work. The review conducted revealed that most existing DARM systems had problems with the database size and memory challenges, which made them ineffective for real-life scenarios. This work therefore addressed a number of these challenges by tapping into the power of mobile agents, which was programmed and deployed to carry out the DARM tasks. Results obtained from the experiments performed in this work showed PEMA outperformed other benchmark algorithms in this area like Apriori, AprioriTFP and FP-Growth when tested on real datasets in terms of the total response time of the algorithms. Further results obtained from comparing PEMA with existing DARM algorithms also showed that it performed better in terms of response time and sizes of messages exchanged. PEMA adopted the logic of vertical partitioning the very large data into distributed data sites and performed better than methods such as CD and DDN, which used horizontal segmentation for same data. We can conclude from our results that for large databases, mobile agent-based vertically partitioning of datasets will always yield better results than horizontally partitioned datasets in terms of response time and communication overhead. It was also observed from the results that PEMA scaled better than the existing methods as the size of updated databases becomes bigger and bigger. Finally in PEMA, scalability and efficiency of this method were also properly addressed. In addition, effective distribution and parallelization of DARM tasks was attained while improved flexibility and usability were also achieved and finally response time and communication overhead of distributed association rule mining tasks were generally reduced. Finally, since PEMA assumed a homogeneous DARM environment, one interesting direction to also further this work is to design an improved PEMA that will work with multiple data schemata residing in heterogeneous environments. Future work will also examine distributed mining and security issues in outsourcing of DARM tasks in the cloud computing environment.

References

[1] Shakshuki E. Methodology for evaluating agent toolkits. In: Proceedings of the knowledge-based intelligent information and engineering systems. Springer; 2005. p. 941–9, 4694/2010.

[2] Kaosar MG, Paulet R, Yi X. Fully homomorphic encryption based two-party association rule mining. Elsevier J: Data Knowledge Eng 2015;76–78:1–15.

[3] Adhikari A, Adhikari J, Pedrycz W. Data analysis and pattern recognition in multiple databases. Intelligent systems reference library 61, @ Springer International Publishing, Switzerland; 2014. p. 21–42.

[4] Saravanan S, Christopher T. A study on milestones of association rule mining algorithms in large databases. Int J Comput Appl 2012;47(3):12–9.

[5] Gharib TF, Nassar H, Taha M, Abraham A. An efficient algorithm for incremental mining of temporal association rules. Data Knowledge Eng 2010;69:800–15.

[6] Ahangar YB, Motameni H, Varzi RA. Identification of Mazandaran telecommunication company fixed phone subscribers using H-means and W–K-means algorithm. Int J Mech Electr, Comput Technol 2013;3(7):1068–79.

[7] Albashiri KA. EMADS: an investigation into the issues of multi-agent data mining. PhD Thesis, The University of Liverpool, Liverpool L69 3BX, United Kingdom; 2010.

[8] Ariwa FI, Mohamed BS, Mohamed MM. Informatization and E-business model application for distributed data mining using mobile agents. International Conference WWW/Internet2003; 2003.

[9] Savasere A, Omiecinski E, Navathe S. An efficient algorithm for mining association rules in large databases. In: Proceedings of the 21st conference on very large databases (VLDB'95), Zurich, Switzerland; 1995. p. 432–44.

[10] Albashiri KA, Coenen F, Leng P. An investigation into the issues of multi-agent data mining. In: Bouca D, Gafagnao A, editors. Agent based computing. Nova Science Publishers; 2010, 978-1-60876-684-0.

[11] Coenen F, Leng P. Optimising association rule algorithms using itemset ordering. In: Proceedings of the AI Conference, Research and Development in Intelligent Systems XVIII. Springer; 2006. p. 53–66.

[12] Pudi V, Haritsa J. ARMOR: Association rule mining based on Oracle. In: ICDM workshop on frequent itemset mining implementations, Florida, USA; 2003.

[13] Nguyen S, Orlowska M. Improvements in the data partitioning approach for frequent itemsets mining. In: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 05). Springer; 2005. p. 625–33.

[14] McConnell S, Skillicorn D. Building predictors from vertically distributed data. In: Proceedings of the 2004 conference of the centre for advanced studies conference on collaborative research 04-07. Markham, Ontario, Canada; 2004. p. 150–62.

[15] Albashiri KA. Agent based data distribution for parallel association rule mining. Int J Comput 2014;8:24–32.

[16] Paul S. An optimized distributed association rule mining algorithm in parallel and distributed data mining with xml data for improved response time. Int J Comput Sci Inform Technol 2010;2(2).

[17] Zaki M. Parallel and distributed association mining: an introduction. In: Proceedings of the large-scale parallel data mining, lecture notes in artificial intelligence 1759. Berlin, Germany: Springer-Verlag; 2000. p. 1–23.

[18] Cheung DW, Xiao Y. Effect of data skewness in parallel mining of association rules. In: Proceedings of the 2nd Pacific-Asia conference on knowledge discovery and data mining, Melbourne, Australia; 1999. p. 48–60.

[19] Agrawal R, Shafer J. Parallel mining of association rules. Proceedings of the IEEE transactions on knowledge and data engineering 1996;8(6):962–9.

[20] Deypir M, Sadreddini MH. Distributed association rules mining using nonderivable frequent patterns. Iran J Sci Technol, Trans B: Eng 2009;33(B6):511–26.

[21] Wooldridge M. An introduction to multi-agent systems. Chichester, United Kingdom: John Wiley and Sons; 2009.

[22] Ogunde AO, Folorunso O, Sodiya AS, Oguntuase JA. An intelligent multi-agent architecture for distributed association rule mining. Comput Inform Syst J 2011;15(2).

[23] Frank A, Asuncion A. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science; 2010. <http://archive.ics.uci.edu/ml>.

[24] Coenen F. LUCS-KDD ARM DN Software. Department of Computer Science, The University of Liverpool, UK; 2003. <http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN_ARM/>.

[25] Venkatesan N, Ramaraj E. High performance bit search mining technique. Int J Comput Appl 2011;14(2):0975–8887.

[26] Ashrafi MZ, Taniar D, Smith K. ODAM. An optimized distributed association rule mining algorithm. IEEE distributed systems online 1541–4922, published by the IEEE computer society; 2004, vol. 5, no. 3.

[27] Kaosar MD, Xu Z, Yi X. Distributed association rule mining with minimum communication overhead. In: Proceedings of the 8th Australasian Data Mining Conference (AusDM'09), School of Engineering and Science, Victoria University, Australia; 2009.