ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

Data Partitioning and Association Rule Mining Using a Multi-Agent System

Dr. Kamal Ali Albashiri
Department of Computer Science, Faculty of Accounting, Algabel Algarbi University
Maydan Aljazer, P.O.Box 1531, Tripoli, Libya

Abstract*— This paper explores and demonstrates (by experiment) the capabilities of Multi-Agent Data Mining (MADM) System in the context of parallel and distributed Data Mining (DM). The exploration is conducted by considering a specific parallel/distributed DM scenario, namely data (vertical/horizontal) partitioning to achieve parallel/distributed ARM. To facilitate the partitioning a compressed set enumeration tree data structure (the T-tree) is used together with an associated ARM algorithm (Apriori-T). The aim of the scenario is to demonstrate that the MADM vision is capable of exploiting the benefits of parallel computing; particularly parallel query processing and parallel data accessing. In addition the approach described offers significant advantages with respect to computational efficiency when compared to alternative mechanisms for (a) dividing the input data between processors (agents) and (b) achieving distributed/parallel ARM.

***Index Terms*— Association Rule Mining, Frequent Itemsets, Meta Mining, Multi-Agent Data Mining.**

I. INTRODUCTION

A common feature of most DM tasks is that they are resource intensive and operate on large sets of data. Data sources measured in gigabytes or terabytes are quite common in DM. This has called for fast DM algorithms that can mine very large databases in a reasonable amount of time. However, despite the many algorithmic improvements proposed in many serial algorithms, the large size and dimensionality of many databases makes the DM of such databases too slow and too big to be processed using a single process. There is therefore a growing need to develop efficient parallel DM algorithms that can run on distributed systems. There are several ways in which data distribution can occur, and these require different approaches to model construction, including:

A. Horizontal Data Distribution

The most straight forward form of distribution is horizontal partitioning, in which different records are collected at different sites, but each record contains all of the attributes for the object it describes. This is the most common and natural way in which data may be distributed. For example, a multinational company deals with customers in several countries, collecting data about different customers in each country. It may want to understand its customers worldwide in order to construct a global advertising campaign.

B. Vertical Data Distribution

The second form of distribution is vertical partitioning, in which different attributes of the same set of records are collected at different sites. Each site collects the values of one or more attributes for each record and so, in a sense, each site has a different view of the data. For example, a credit-card company may collect data about transactions by the same customer in different countries and may want to treat the transactions in different countries as different aspects of the customers total card usage. Vertically partitioned data is still rare, but it is becoming more common and important [9]. This paper addresses a generic MADM scenario, that of distributed/parallel DM. This scenario assumes an end user who owns a large data set and wishes to obtain DM results but lacks the required resources (i.e. processors and memory). The data set is partitioned into horizontal or vertical partitions that can be distributed among a number of processors (agents) and independently processed, to identify local itemsets, on each process.
In the exploration of the applicability of MADM to parallel/distributed ARM, the two data partitioning approaches, based on the Apriori algorithm, are described and their performance evaluated as indicated above. Recall that DATA-HS (Horizontal Segmentation) makes use of a horizontal partitioning of the data. The data is apportioned amongst a number of data agents, typically by horizontally segmenting the dataset into sets of records.
DATA-VP makes use of a vertical partitioning approach to distributing the input dataset over the available number of DM (worker) agents. To facilitate the vertical data partitioning the tree data structure, described in [7], is again used together with the Apriori-T ARM algorithm [6]. Using both approaches each partition can be mined in

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

isolation, while at the same time taking into account the possibility of the existence of frequent itemsets dispersed across two or more partitions. In the first approach, DATA-HS, the scenario complements the meta ARM scenario described in Albashiri et al. [3].

The rest of the paper is organized as follows. Section 2 provides an overview of the field of MADM. Data partitioning is introduced in Section 3. Data partitioning may be achieved in either a horizontal or vertical manner. In Section 4 a parallel/distributed task with Data Horizontal Segmentation (DATA-HS) algorithm is described. Before describing the vertical approach the Apriori-T algorithm is briefly described in Section 5. The parallel/distributed task with Data Vertical Partitioning (DATA-VP) algorithm (which is founded on Apriori-T) is then described in Section 6. The DATA-VP MADM task architecture and network configuration is presented in Section 7. Experimentation and Analysis, comparing the operations of DATA-HS and DATA-VP, is then presented in Section 8. Discussion of how this scenario addresses the goal of this paper is presented in Section 9. Finally a summary and conclusion is given in Section 10.

II. BACKGROND AND LITERATURE REVIEW

This section briefly presents a review of the current research relating to Multi-Agent Data Mining (MADM). It provides an overview of the theoretical background of the research discipline, identifying the approaches adopted, and discusses the benefits and challenges posed.

During the last two decades, our ability to collect and store data has significantly outpaced our ability to analyze, summarize and extract "knowledge" from this data. The phrase Knowledge Discovery in Databases (KDD) denotes the complex process of identifying valid, novel, potentially useful and ultimately understandable patterns in data [15]. DM refers to a particular step in the KDD process. It consists of particular algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns (models) over the data.

A considerable number of algorithms have been developed to perform DM tasks, from many fields of science [16]. Typical DM tasks are classification (the generation of classifiers which can be used to assign each record of a database to one of a predefined set of classes), clustering (finding groups of database records that are similar according to some user defined metrics) or ARM (determining implication rules for a subset of database record attributes).

Agents and multi-agent systems are an emergent technology that is expected to have a significant impact in realizing the vision of a global and information rich services network to support dynamic discovery and interaction of digital enterprises. Significant work on multi-agent systems has already been done for
more than a decade since agents were first claimed to be the next breakthrough in software development, resulting in powerful multi-agent platforms and innovative e-business applications.

Multi-agent Data Mining (MADM) is concerned with the use of agent and MAS to perform DM activities.

Several systems have been developed for MADM. These systems can be categorized, according to their strategy of learning, into three types:

1. Central-learning, when all the data can be gathered at a central site and a single model built [17].
2. Meta-learning, is the process of automatic induction of correlations between tasks and solving strategies, based on a domain characterization [18].
3. Hybrid-learning is a technique that combines local and remote learning for model building [19].

The most popular task of DM is to find patterns in data that show associations between domain elements. This is generally focused on transactional data, such as a database of purchases at a store. This task is known as Association Rule Mining (ARM), and was first introduced in Agrawal et al. [2]. Association Rules (ARs) identify collections of data attributes that are statistically related in the underlying data.

III. DATA SEGMENTATION AND PARTITIONING

Notwithstanding the extensive work that has been done in the field of ARM, there still remains a need for the development of faster algorithms and alternative heuristics to increase their computational efficiency. Because of the inherent intractability of the fundamental problem, much research effort has been directed at parallel ARM to decrease overall processing times (see [8, 11, 12, 13]), and distributed ARM to support the mining of datasets distributed over a network [4]. The main challenges associated with parallel DM include:

- Minimizing I/O.
- Minimizing synchronization and communication.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

• Effective load balancing.
• Effective data layout (horizontal vs. vertical).
• Good data decomposition.
• Minimizing/avoiding duplication of work.

To allow the data to be mined using a number of cooperating agents the most obvious approach is to allocate different subsets of the data to each agent. There are essentially two fundamental approaches to partitioning/segmenting the data:

1. Horizontal segmentation where the data is divided according to row number.
2. Vertical partitioning where the data is divided according to column number.

Note that in this paper the term partitioning is used to indicate vertical sub-division of data, and segmentation to indicate horizontal sub-division of data.

Horizontal segmentation is in general more straightforward. Assuming a uniform/ homogeneous dataset it is sufficient to divide the number of records by the number of available agents and allocate each resulting segment accordingly.

The most natural method to vertically partition a dataset is to divide the number of columns by the number of available agents so each is allocated an equal number of columns.

Many parallel DM algorithms have been developed based on the Apriori algorithm or variations of the Apriori algorithm. The most common parallel methods are [2, 8]:

• Count Distribution. This method follows a data-parallel strategy and statically partitions the database into horizontal partitions that are independently scanned for the local counts of all candidate itemsets on each process. At the end of each iteration, the local counts are summed across all processes to form the global counts so that frequent itemsets can be identified.
• Data Distribution. The Data Distribution method attempts to utilize the aggregate main memory of parallel machines by partitioning both the database and the candidate itemsets. Since each candidate itemset is counted by only one process, all processes have to exchange database partitions during each iteration in order for each process to get the global counts of the assigned candidate itemsets.
• Candidate Distribution. The Candidate Distribution method also partitions candidate itemsets but selectively replicates, instead of "partitioned-exchanging" the database transactions, so that each process can proceed independently.

Experiments show that the Count Distribution method exhibits better performance and scalability than the other two methods [2]. The steps for the Count Distribution method may be generalized as follows (for distributed-memory multiprocessors):

1. Divide the database evenly into horizontal partitions among all processes.
2. Each process scans its local database partition to collect the local count of each item.
3. All processes exchange and sum up the local counts to get the global counts of all items and find frequent 1-itemsets.
4. Set level k = 2.
5. All processes generate candidate k-itemsets from the mined frequent (k-1)-itemsets.
6. Each process scans its local database partition to collect the local count of each candidate k-itemset.
7. All processes exchange and sum up the local counts into the global counts of all candidate k-itemsets and find frequent k-itemsets among them.
8. Repeat (5) - (8) with k = k + 1 until no more frequent itemsets are found.

In the following sections two MADM tasks, using both vertical partitioning and horizontal segmentation, are introduced. These tasks were implemented using a task wrapper, so that they could be incorporated into the system as task agents.

IV. THE PARALLEL/DISTRIBUTED TASK WITH DATA-HS ALGORITHM

The Data Horizontal Segmentation (DATA-HS) algorithm uses horizontal segmentation, dividing the dataset into segments each containing an equal number of records. ARM in this case involves the generation of a number of T-trees, holding frequent itemsets, one for each segment; and then merging these T-trees to create one global T-tree. As a demonstration of MADM re-usability, this is carried out using the Meta ARM task agent. Further details of this process can be also found in Albashiri et al. [3].

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

Assuming that a data agent representing the large dataset has been launched by a user, the DATA-HS MADM algorithm comprises the following steps:

- User agent requests the task agent to horizontally segment the dataset according to the total number of segments required.
- The task agent assigns and sends each data segment to an interested data agent; if none exist then the task agent launches new data agents.
- Then a meta ARM task is called to obtain the Association Rules (ARs) as described in [3].

V. THE APRIORI-T ALGORITHM

The Apriori-T (Apriori Total) algorithm is an Association Rule Mining (ARM) algorithm [7] that combines the classic Apriori ARM algorithm with the T-tree data structure. As each level is processed, candidates are added as a new level of the T-tree, their support is counted, and those that do not reach the required support threshold pruned. When the algorithm terminates, the T-tree contains only frequent itemsets. The Apriori-T algorithm was developed as part of the more sophisticated ARM algorithm The Apriori-TFP. The Apriori and Apriori-TFP algorithms are described in [7].

At each level, new candidate itemsets of size k are generated from identified frequent k-1 itemsets, using the downward closure property of itemsets, which in turn may necessitate the inspection of neighboring branches in the T-tree to determine if a particular k-1 subset is supported. This process is referred to as X-checking. Note that X-checking adds a computational overhead; offset against the additional effort required to establish whether a candidate k itemset, all of whose k-1 itemsets may not necessarily be supported, is or is not a frequent itemset.

The number of candidate nodes generated during the construction of a T-tree, and consequently the computational effort required, is very much dependent on the distribution of columns within the input data. Best results are produced by ordering the dataset, according to the support counts for the 1-itemsets, so that the most frequent 1-itemsets occur first [5].

VI. THE PARALLEL/DISTRIBUTED TASK WITH (DATA-VP ALGORITHM

The second algorithm considered in the exploration of the applicability of MADM to parallel/distributed ARM is the Data Vertical Partitioning (DATA-VP). The DATA-VP algorithm commences by distributing the input dataset over the available number of workers (DM agents) using a vertical partitioning strategy. Initially the set of single attributes (columns) is split equally between the available workers so that an allocationItemSet (a sequence of single attributes) is defined for each DM agent in terms of a startColNum and endColNum:

AllocationItemSet = {n|startColNum < n · endColNum}

Each DM agent will have its own allocationItemSet which is then used to determine the subset of the input dataset to be considered by the DM agent.

Using its allocationItemSet the task agent will partition the data among workers (DM agents) as follows:

1. Remove all records in the input dataset that do not intersect with the allocationItemSet.
2. From the remaining records remove those attributes whose column number is greater than endColNum. Attributes whose identifiers are less than startColNum cannot be removed because these may still need to be included in the subtree counted by the DM agent.
3. Send the allocated data partition to the corresponding DM agent.

The input dataset distribution procedure, given an allocationItemSet, can be summarized as follows:

Table 1: Dataset Example

```
ʉ records ϵ input data
if (record ∩ allocationItemSet ≡ true)
record = {n|n ϵ record, n ≤ endColNum}  else
delete record
```

TID	ItemSet
1	acf
2	b
3	ace
4	bd
5	ae
6	abc
7	d
8	ab
9	c
10	abd

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

As an example, the ordered data set in Table 1 has items with 6 attributes, a, b, c, d, e and f. Assuming three worker agents are participating, the above partitioning process will result in three dataset partitions, with allocationItemSets {a, b}, {c, d} and {e, f}. Application of the above algorithm will create partitions as follows (but note that the empty sets, here shown for clarity, will in fact not be included in the partitions):

Partition 1 (a to b): {{*a*}, {*b*}, {*a*}, {*b*}, {*a*}, {*a, b*}, {}, {*a, b*}, {}, {*a, b*}}
Partition 2 (c to d): {{a, c}, {}, {a, c}, {b, d}, {}, {a, b, c}, {d}, {}, {c},{a, b, d}}
Partition 3 (e to f): {{*a, c, f*}, {}, {*a, c, e*}, {}, {*a, e*}, {}, {}, {}, {}}

Once partitioning is complete each partition can be mined, using the Apriori-T algorithm, in isolation while at the same time taking into account the possibility of the existence of frequent itemsets dispersed across two or more partitions.
Figure 1 shows the resulting sub T-trees assuming all combinations represented by each partition are supported. Note that because the input dataset is ordered according to the frequency of 1-itemsets the size of the individual partitioned sets does not necessarily increase as the endColNum approaches N (the number of columns in the input dataset); in the later partitions, the lower frequency leads to more records being eliminated. Thus the computational effort required to process each partition is roughly balanced.

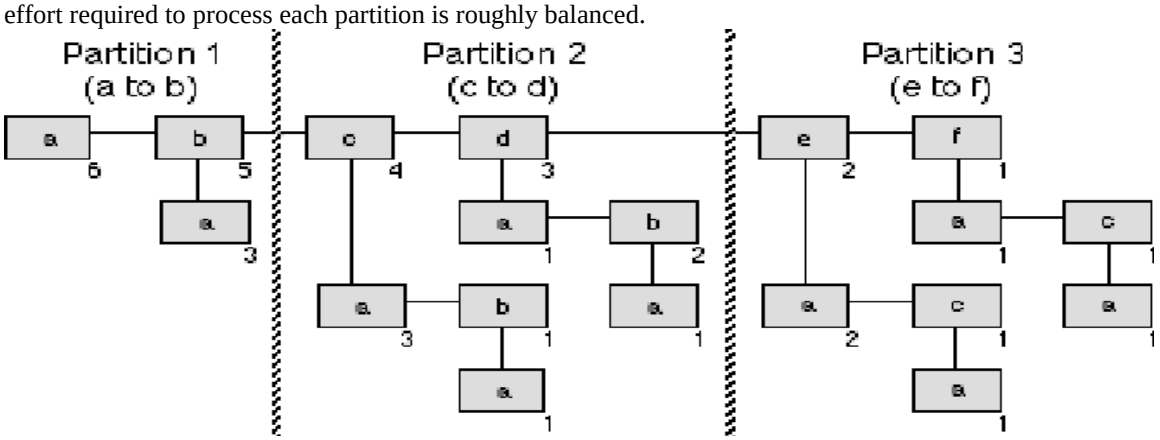


Fig 1: Vertical Partitioning of a T-tree Example [6]

The DATA-VP MADM task can thus be summarized as follows:
1. A task agent starts a number of workers (DM agents); these will be referred to as *workers*.
2. The task agent determines the division of allocationItemSet according to the total number of available workers (agents) and transmits this information to them.
3. The task agent transmits the allocated partition of the data (calculated as described above) to each worker.
4. Each worker then generates a T-tree for its allocated partition (a sub T-tree of the final T-tree).
5. On completion each DM (worker) agent transmits its partition of the T-tree to the task agent which are then merged into a single global T-tree (the final T-tree ready for the next stage in the ARM process, rule generation).
The local T-tree generation process begins with a top-level "tree" comprising only those 1-itemsets included in each worker (DM agent) allocationItemSet.
The DM agent will then generate the candidate 2-itemsets that belong in its sub (local) T-tree. These will comprise all the possible pairings between each element in the allocationItemSet and the lexicographically preceding attributes of those elements (see Figure 1). The support values for the candidate 2-itemsets are then determined and the sets pruned to leave only frequent 2-itemsets. Candidate sets for the third level are then generated. Again, no attributes from succeeding allocationItemSet are considered, but the possible candidates will, in general, have subsets which are contained in preceding allocationItemSet and which, therefore, are being counted by some other DM agents. To avoid the overhead involved in the X-checking process, described in Section 4, which in this case would require message-passing between the DM agents concerned, X-checking does not take place. Instead, the DM agent will generate its candidates assuming, where necessary, that any subsets outside its local T-tree are frequent.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

VII. DATA-VP TASK ARCHITECTURE AND NETWORK CONFIGURATION

The DATA-VP task architecture shown in Figure 2 assumes the availability of at least one worker (DM agent), preferably more. Figure 2 shows the assumed distribution of agents and shared data across the network. The figure also shows the house-keeping JADE agents (AMS and DF) through which agents find each other. JADE (Java Agent Development Environment) [14] is a multi-agent platform which this system is implemented in.

A. Messaging

Parallel/distributed ARM tends to entail much exchange of data messaging as the task proceeds. Messaging represents a significant computational overhead, in some cases outweighing any other advantage gained. Usually the number of messages sent and the size of the content of the message are significant factors affecting performance. It is therefore expedient, in the context of the techniques described here, to minimize the number of messages that are required to be sent as well as their size.

The technique described here is One-to-Many approach, where only the task agent can send/receive messages to/from DM agents. This involves fewer operations, although, the significance of this advantage decreases as the number of agents used increases.

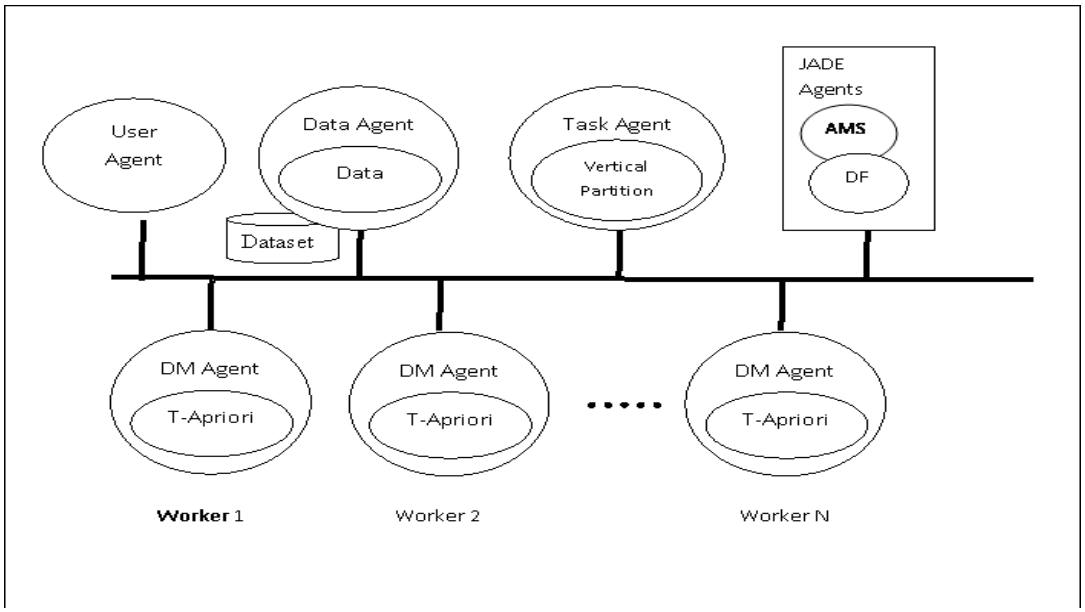


Fig 2: Parallel/Distributed ARM Model for DATA-VP Task Architecture

VIII. EXPERIMENTATION AND ANALYSIS

To evaluate the two approaches, in the context of the MADM vision, a number of experiments were conducted. These are described and analyzed in this section.

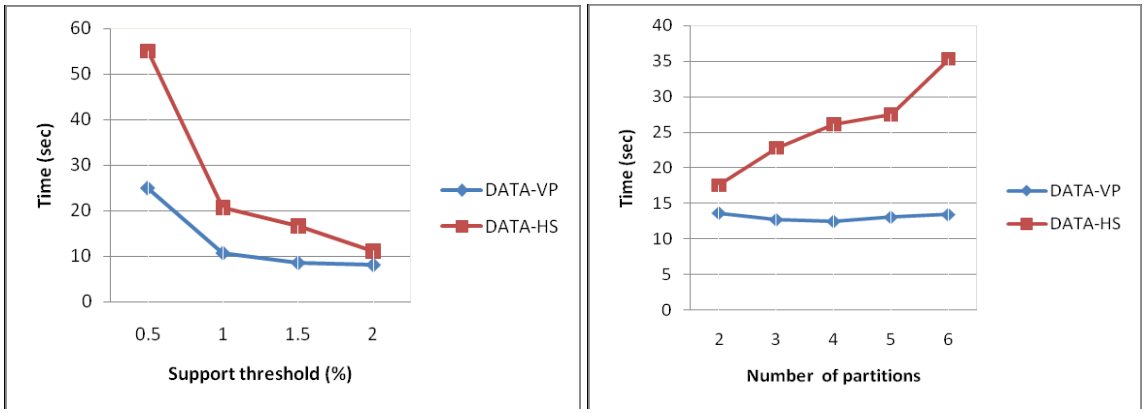
The experiments presented here used up to six data partitions and two artificial datasets: (i) T20.D100K.N250.num, and (ii) T20.D500K.N500.num where T = 20 (average number of items per transactions), D = 100K or D = 500K (Number of transactions), and N = 500 or N = 250 (Number of items) are used. The datasets were generated using the IBM Quest generator used in Agrawal and Srikant [1].

As noted above the most significant overhead of any parallel/distributed system is the number and size of messages sent and received between agents. For the DATA-VP approach, the number of messages sent is independent of the number of levels in the T-tree; communication takes place only at the end of the tree construction. DATA-VP passes entire pruned sub (local) T-tree branches. Therefore, DATA-VP has a clear advantage in terms of the number of messages sent.

Figure 3 and Figure 4 show the effect of increasing the number of data partitions with respect to a range of support thresholds. As shown in Figure 3 the DATA-VP algorithm shows better performance compared to the DATA-HS algorithm. This is largely due to the smaller size of the dataset and the T-tree data structure which: (i) facilitates

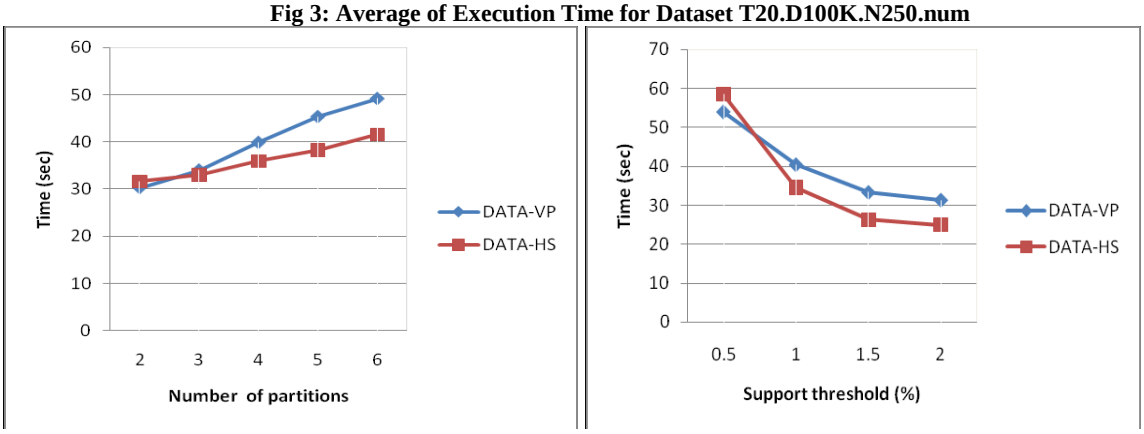
ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

vertical distribution of the input dataset, and (ii) readily lends itself to parallelization/distribution. However, when the data size is increased as in the second experiment, and further DM (worker) agents are added (increasing the number of data partitions), the results shown
in Figure 4, show that the increasing overhead of messaging size outweighs any gain from using additional agents, so that parallelization/distribution becomes counter productive. Therefore DATA-HS showed better performance from the addition of further data agents compared to the DATA-VP approach.



(a)Number of Data Partitions **(b) Support Threshold**

Fig 3: Average of Execution Time for Dataset T20.D100K.N250.num



(a) Number of Data Partitions **(b) Support Threshold**

Fig 4: Average of Execution Time for Dataset T20.D500K.N500.num

IX. DISCUSSION

MADM can be viewed as an effective distributed and parallel environment where the constituent agents function autonomously and (occasionally) exchange information with each other. The MADM system is designed with asynchronous, distributed communication protocols that enable the participating agents to operate independently and collaborate with other peer agents as necessary, thus eliminating centralized control and synchronization barriers.

Distributed and parallel DM can improve both efficiency and scalability first by executing the DM processes in parallel improving the run-time efficiency and second, by applying the DM processes on smaller subsets of data that are properly partitioned and distributed to fit in main memory (a data reduction technique).

The scenario, described in this paper, demonstrated that MADM provides suitable mechanisms for exploiting the benefits of parallel computing; particularly parallel data processing. The scenario also demonstrated that MADM is suitable for re-usability and illustrated how it is supported by re-employing the meta ARM task agent, described in the previous paper, with the DATA-HS task.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

X. SUMMARY AND CONCLUSION

In this paper a MADM method for parallel/distributed ARM has been described so as to explore the MADM issues of scalability and re-usability. Scalability is explored by parallel processing of the data and re-usability is explored by reemploying the Meta ARM task agent with the DATA-HS task.

The solution to the scenario considered in this paper made use of a vertical data partitioning or a horizontal data segmentation technique to distribute the input data amongst a number of agents. In the horizontal data segmentation (DATA-HS) method, the dataset was simply divided into segments each comprising an equal number of records. Each segment was then assigned to a data agent that allowed for using the meta ARM task when employed on a MADM system. Each DM agent then used its local data agent to generate a complete local T-tree for its allocated segment. Finally, the local T-trees were collated into a single tree which contained the overall frequent itemsets. The proposed vertical partitioning (DATA-VP) was facilitated by the T-tree data structure, and an associated mining algorithm (Apriori-T), that allowed for computationally effective parallel/distributed ARM when employed on the MADM system.

The reported experimental results showed that the data partitioning methods described are extremely effective in limiting the maximal memory requirements of the algorithm, while their execution time scale only slowly and linearly with increasing data dimensions. Their overall performance, both in execution time and especially in memory requirements has brought significant improvement.

REFERENCES

[1] R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger. The Quest Data Mining System. In Proceedings of the 2nd International Conference Knowledge Discovery and Data Mining, (KDD), 1996.

[2] R. Agrawal and J. Shafer. Parallel mining of association rules. In Proceedings of the IEEE Transactions on Knowledge and Data Engineering 8(6), pages (962-969), 1996.

[3] K. A. Albashiri, F. Coenen, and P. Leng. EMADS: An Extendible Multi-Agent Data Miner, volume XXIII, pages (263-276). Research and Development in Intelligent Systems, AI, Springer, London, England, 2008.

[4] D. Cheung and Y. Xiao. Effect of Data Distribution in Parallel Mining of Associations. In Proceedings of the Data Mining and Knowledge Discovery 3(3), pages (291-314), 1999.

[5] F. Coenen and P. Leng. Optimising Association Rule Algorithms Using Itemset Ordering. In Proceedings of the AI Conference, Research and Development in Intelligent Systems XVIII, Springer, pages (53-66), 2001.

[6] F. Coenen, P. Leng, and S. Ahmed. T-Trees, Vertical Partitioning, and Distributed Association Rule Mining. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Florida, eds. X Wu, A Tuzhilin and J Shavlik: IEEE Press, pages (513-516), 2003.

[7] F. Coenen, P. Leng, and G. Goulbourne. Tree Structures for Mining Association Rules, volume 8(1), pages (25-51). In the Journal of Data Mining and Knowledge Discovery, 2004.

[8] E. Han, G. Karypis, and V. Kumar. Scalable Parallel Data Mining for Association Rules. In Proceedings of the ACM(Association for Computer Machinery)- Special Interest Group on Management of Data (SIGMOD), International Conference on Management of Data, ACM Press, pages (277-288), 1997.

[9] S. McConnell and D. Skillicorn. 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, pages (150-162), 2004.

[10] D. Michie, D. Spiegel halter, and C. Taylor. Machine Learning, Neural and Statical Classification. In Ellis Horwood Series in Artificial Intelligence, New York, 1994.

[11] S. Parthasarathy, M. Zaki, and W. Li. Memory Placement Techniques for Parallel Association Mining. In Proceedings of the 4th International Conference on Knowledge Discovery in Databases (KDD), AAAI Press, pages (304-308), 1998.

[12] T. Shintani and M. Kitsuregawa. Hash Based Parallel Algorithms for Mining Association Rules. In Proceedings of the 4th International Conference on Parallel and Distributed Information Systems, (PIDS), IEEE Computer Society Press, pages (19-30), 1996.

[13] M. Tamura and M. Kitsuregawa. Dynamic Load Balancing for Parallel Association Rule Mining on Heterogeneous PC Cluster Systems. In Proceedings of the 25th Very Large Data Bases (VLDB) Conference, Morgan Kaufman, pages (162-173), 1999.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 5, September 2013

[14] F. Tellifemine, A. Poggi, and G. Rimassi. JADE: A FIPA-Compliant agent framework. In Proceedings the Practical Applications of Intelligent Agents and Multi-Agents, pages (97-108), 1999. http://www.jade.tilab.com.

[15] G. Piatetsky-Shapiro, U. Fayyad, P. Smyth, and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. the Association for the Advancement of Artificial Intelligence (AAAI) Press/MIT Press, 1996.

[16] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg.

[17] Top 10 Algorithms in Data Mining, Knowledge and Information Systems, volume 14, pages (1-37). Springer-Verlag, London Limited, 2008.

[18] H. Xargupta, I. Hamzaoglu, and B. Stafford. Scalable, Distributed Data Mining Using an Agent Based Architecture. Proceedings of Knowledge Discovery and Data Mining, AAAI Press, pages (211-214), 1997.

[19] J. A. Yota, A. F. Gmez-Skarmeta, M. Valds, and A. Padilla. Metala. A meta-learning architecture. Fuzzy Days, pages (688-698), 2001.

[20] A. Zurinsky and R. Grossman. A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies. In KDD Workshop on Distributed Data Mining, 2000.

AUTHOR BIOGRAPHY

Kamal Albashiri, PhD in computer science, is a university lecturer at the Faculty of Accounting, Algabel Algarbi University, Gharan, Libya. He has published many research papers and book chapters in data mining and multi-agent, his research work focused on investigation the issues of multi-agent and data mining integration.