ISSN:0975-9646

Extensible Multi Agent System for Heterogeneous Database Unification Using Parallel Task and Segmentation Algorithm

[1] A.Ramar [2] M.Elamparuthi

[1]Research Scholar, Sree Saraswathi Thyagaraja College, Pollachi.
[2]Assistant Professor, Department of Computer Science, Sree Saraswathi Thyagaraja College, Pollachi.

Abstract – Knowledge Discovery Databases (KDD) has evolved to become a well established technology that has many commercial applications. It encompasses sub fields such as classification, clustering, and rule mining. However, it still poses many challenges to the research community. New methodologies are needed in order to mine more interesting and specific information from larger datasets. New techniques are needed to harmonize more effectively the steps of the Knowledge Discovery in Databases process. New solutions are required to manage the complex and heterogeneous sources of information that are today available for the analysis. Knowledge Discovery Databases is concerned with the extraction of hidden knowledge from data. Extendibility also implies flexibility, and Multi Agent Data Mining should support the inclusion of existing as well as future Data Mining technology (otherwise the Multi Agent Data Mining will rapidly be rendered inadequate and therefore obsolete). Essentially adding a new data source or Data Mining technique to a Multi Agent Data Mining should equate to the introduction of new agents.

Keywords: Knowledge Discovery in Database, Multi Agent Systems, Extendible Multi Agent Data mining System.

1. INTRODUCTION

In general, extendibility can be defined as the ease with which software can be modified to adapt to new requirements, or changes in existing requirements. In the context of Multi Agent Data Mining extendibility is defined as the ease with which new data sources and new Data Mining techniques can be introduced to the framework. Extendibility also implies flexibility, and Multi Agent Data Mining should support the inclusion of existing as well as future Data Mining technology (otherwise the Multi Agent Data Mining will rapidly be rendered inadequate and therefore obsolete). Essentially adding a new data source or Data Mining technique to a Multi Agent Data Mining should equate to the introduction of new agents.

There are several reasons to ask for extendibility. For example, a piece of existing software may do nearly the right thing, but needs some addition to meet a new requirement. Or it may do the right thing, but needs to be wrapped up to provide easier handling and integration into other software components. Or, as in the Extendible Multi Agent Data Mining System case, the need to add new functionalities which make use of the services already provided. There are several methods in the literature available to provide extendibility, including:

Software Wrappers: Software wrapping refers to a re-engineering technique that surrounds a software component or tool with a new software layer to hide the internal code and the logic of the component or tool and to supply modern interfaces. One example of software wrapping is the reuse of legacy software in modern applications, where the unwanted complexity of the old software is hidden to the applications. Software wrapping re-moves the mismatch between the interfaces exported by a legacy software "artifact" and the interfaces required by the current software program. The advantage of wrapping is that the added component becomes part of the system without changing the content of the component. Using wrappers can greatly reduce the amount of work required to build new functionalities.

Libraries: Libraries are an alternative mechanism for providing a reusable abstraction of code. Libraries also known as an archives, consists of a set of routines which are copied into a target application by the compiler, or linker producing object files and a stand-alone executable file. This process is known as a build of the target application. However, unlike wrappers, a library leaves control in the hands of the programmer. Since they do not provide any flow control by themselves, complex interactions must be built from scratch.

Plug-ins: Plug-ins is miniature programs that "plug into" a host program for additional functionality. Plug-ins allows a third-party application to be used within the host program, acting as a kind of liaison or bridge. Plug-ins is a widely-used approach to provide application extendibility. For example, makers of popular software such as Adobe and Mozilla provide plug-in architectures for their applications. Plug-ins is required to interact with the host application through an API which provides access to a subset of the host's functionality. Unfortunately, this usually means plug-ins can only provide restricted functionality to enhance the host application.

Dynamic scripting: Dynamic scripting to influence functionality is an-other way to extend applications. Dynamic scripting is used within single applications as extension/customization tools that allow users to customize, connect, and control the components of the application. Scripts are distinct from the core code of the application, as they are usually written in a different language and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code. For example, the EMACS framework allows users to define new functionality on the fly or even redefine existing functionality. This ability to redefine existing behaviour makes the EMACS extendibility approach much like that of plug-ins. Furthermore the core implementation retains control over program flow. This approach is beneficial because it can be interpreted at run-time instead of compile time. However, this carries the performance penalties associated with interpreted code.

2. Extendibility In Extendible Multi Agent Data Mining System

To ensure generality and extendibility, Extendible Multi Agent Data Mining System is designed using object-oriented methods and is implemented independently of any particular machine learning program or Data Mining task. Extendible Multi Agent Data Mining System extendible capabilities provide the means to easily incorporate any new Data Mining algorithm/task or data sources. The independence of Extendible Multi Agent Data Mining System from any particular Data Mining method, in conjunction with the object oriented design, ensure the framework's capability to incorporate and use new Data Mining algorithms/tasks or data sources. Data Mining techniques and data sources can be embedded within appropriate wrappers to be introduced into the Extendible Multi Agent Data Mining System and subsequently become an Extendible Multi Agent Data Mining System agent. Introducing a new technique requires the use of the appropriate wrapper to encapsulate the algorithm/technique within an object that adheres to the minimal wrapper interface. In fact, most of the existing implemented Data Mining algorithms have similar interfaces already.
Extendible Multi Agent Data Mining System wrappers define the abstract methods of the parent classes and are responsible for invoking the executables of these algorithms. This extendibility characteristic makes Extendible Multi Agent Data Mining System an extendible Data Mining facility. Evaluation of Extendible Multi Agent Data Mining System has shown that, given an appropriate wrapper, existing data mining software can be very easily packaged to become an Extendible Multi Agent Data Mining System agent. This feature of Extendible Multi Agent Data Mining System was investigated using a number of different scenarios. These scenarios are discussed in later chapters.

2.1 WRAPPERS

As Extendible Multi Agent Data Mining System agents are not intended as replacements for Data Mining algorithms or techniques, the agents must be able to interact with the algorithms or techniques. Generally speaking there are at least three possible approaches to incorporating Data Mining software. The software could be rewritten to an appropriate form, but this is a costly approach. Alternatively a separate piece of software could be employed to act as an interpreter between the agent communication language and the native protocol of the Data Mining software. Or thirdly, the wrapper technique could be used to augment the legacy program with code that enables it to communicate using the inter-agent language. As pointed out earlier, Extendible Multi Agent Data Mining System uses the latter approach. The term wrapper in the context of computer science has a number of connotations (for example "driver wrappers", "TCP wrappers" and the "Java wrapper" classes). In the context of Multi Agent System the term is used to describe a mechanism for allowing existing software systems to be incorporated into Multi Agent System. The wrapper "agentifies" an existing application by encapsulating its implementation. The wrapper manages the states of the application, invoking the application when necessary.
Extendible Multi Agent Data Mining System wrappers are used to "wrap" up Data Mining artifacts so that they become Extendible Multi Agent Data Mining System agents and can communicate with other agents within Extendible Multi Agent Data Mining System. As such Extendible Multi Agent Data Mining System wrappers can be viewed as agents in their own right that are subsumed once they have been integrated with data or tools to become Data Mining agents. The wrappers essentially provide an application interface to Extendible Multi Agent Data Mining System that has to be implemented by the end user, although this has been designed to be a fairly trivial operation. Extendible Multi Agent Data Mining System supports three broad categories of wrapper as shown in Figure –1:

- Data wrapper.
- Data Mining wrapper.
- Task wrapper.

The first is used to create data agents, the second to create Data Mining agents, and the third to create task agents. Figure-1 illustrates the broad "agentification" process for the three different kinds of agent. The figure should be interpreted as follows:
(i) Data wrapper usage is facilitated by a GUI and normally requires no programming.
(ii) DM wrapper usage requires extending and implementing a Java interface.
(iii) Task wrapper usage requires extending and implementing a Java class.

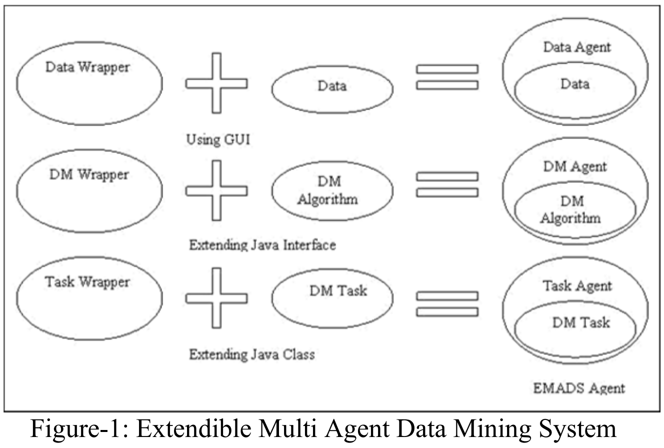


Figure-1: Extendible Multi Agent Data Mining System Wrappers

Data Wrappers: The current version of Extendible Multi Agent Data Mining System assumes well-defined schemata and datasets as to avoid issues of data heterogeneity. In the context of Extendible Multi Agent Data Mining System, a data source is a single text file containing data records (one line per record); where each record comprises a set of, typically comma or space separated, alpha-numeric values that subscribe to some attribute schema. This is a fairly standard tabular data format used throughout the Knowledge Discovery Database community. Knowledge Discovery Database has traditionally been concerned with tabular data sets, reflecting the strong association between Data Mining and relational databases. Data Mining algorithms have generally been developed with respect only to tabular data. A tabular format offers the advantage of easy integration with other tabular data sets. Data wrappers are therefore used to "wrap" a data source and consequently create a data agent. Conceptually the data wrap-per provides the interface between the data source and the rest of the framework. Broadly a data wrapper holds: (i) the location (file path) of a data source, so that it can be accessed by other agents; and (ii) meta information about the data (e.g. data type, number of classes). To assist end users, in the application of a data wrapper to their data, a data wrapper GUI was developed. As described previously, once created, the data agent announces itself to the Extendible Multi Agent Data Mining System facilitator (broker) agent as a consequence of which it becomes available to all Extendible Multi Agent Data Mining System users. In the case of non-standard data, then the user can use the wrapper to have the function of reformatting the data to meet the requirements of Extendible Multi Agent Data Mining System.

Data Mining Wrappers: Data Mining wrappers are used to "wrap" up Data Mining software algorithms and to create DM agents. Generally the algorithms to be wrapped will be Data Mining techniques of various kinds (classifiers, clusters, association rule miners, etc.) Unlike data wrappers, Data Mining wrappers are not supported by a GUI facility to aid their usage; instead Extendible Multi Agent Data Mining System developers are expected to encode the wrappers themselves. However, the wrapper provides most of the required functionalities which makes the en-coding to be a straight forward process.

Task Wrappers: It is intended that the framework will incorporate a substantial number of different Data Mining technique wrappers each defined by the nature of the desired I/O which in turn will be informed by the nature of the generic DM tasks that it is desirable for Extendible Multi Agent Data Mining System to be able to perform. Thus, Extendible Multi Agent Data Mining System developers are expected to encode most of the operations of the task wrappers themselves. Again, the design of the task wrapper is such that encoding of operations is not a significant process.

Extendible Multi Agent Data Mining System approach offers a simple way to incorporate existing methods and data sources into a Multi Agent Data Mining framework and how new Data Mining tasks are wrapped and incorporated. This section looks at what components (agents) are affected when new requirements are added and the extensibility of those components. In the current architecture of Extendible Multi Agent Data Mining System, a single data agent controls each separate data source. The data agent in-turn encapsulates and relies on an instance of an agent class. Thus, in order to bring a new data source into the system, a new data agent must be instantiated. It must also register with the system to ensure the other agents are aware of its existence and the data source can be utilized to fulfill system goals.

In the case of data agents, there is no code required to create a new data agent and introduce it to Extendible Multi Agent Data Mining System. This is achieved through the data wrapper graphical user interface (GUI). The user must use the GUI to refer to the data source location (file path). Then the wrapper creates a new data agent that will represent the data source as it is introduced to the system.

In order to bring a new Data Mining algorithm into the Extendible Multi Agent Data Mining System, a new Data Mining agent must be instantiated with the two required components, namely a Data Mining wrapper and Data Mining algorithm. The new Data Mining agent must also be registered with the system to ensure that other agents are aware of its existence and so that the Data Mining algorithm can be utilized to fulfill Data Mining requests.

The independence of the Data Mining technique comes from the fact that the wrapper agents act only as interfaces, to the Data Mining algorithms, to receive requests and pass back results. Because of this, the only code required to create a new Data Mining agent and "introduce" a new Data Mining algorithm becomes a one line instantiation as shown below:

```
<Algorithm Class Name> instanceName = new <Algorithm Class Name>();
```

The Data Mining wrapper contains all code required to interface with the Data Mining algorithm. In order to ensure all new Data Mining algorithms can communicate in the system, an abstract interface was created. Any new Data Mining algorithm class must extend this abstract interface and implement the abstract methods it includes.

3. Data Segmentation And Partitioning

Notwithstanding the extensive work that has been done in the field of Association Rule Mining, 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 Association Rule Mining to decrease overall processing times and distributed Association Rule Mining to support the mining of datasets distributed over a network. The main challenges associated with parallel Data Mining include:

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

(i) Horizontal segmentation where the data is divided according to row number.

(ii) Vertical partitioning where the data is divided according to column number.

Note that in this chapter 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 Data Mining algorithms have been developed based on the Apriori algorithm or variations of the Apriori algorithm. The most common parallel methods are:

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 partition 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 “partition-and-exchanging” the database transactions, so that each process can proceed independently.

The steps for the Count Distribution method may be generalized as follows (for distributed memory multiprocessors):

Step-1: Divide the database evenly into horizontal partitions among all processes.

Step-2: Each process scans its local database partition to collect the local count of each item.

Step-3: All processes exchange and sum up the local counts to get the global counts of all items and find frequent 1-itemsets.

Step-4: Set level k = 2.

Step-5: All processes generate candidate k-itemsets from the mined frequent (k-1) itemsets.

Step-6: Each process scans its local database partition to collect the local count of each candidate k-itemset.

Step-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.

Step-8: Repeat (5) to (8) with k = k + 1 until no more frequent itemsets are found.

4. The Parallel Task With Horizontal Segmentation (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. Association Rule Mining 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 Multi Agent Data Mining re-usability, this is carried out using the Meta Association Rule Mining task agent which is defined from the previous scenario. Assuming that a data agent representing the large dataset has been launched by a user, the DATA-HS Multi Agent Data Mining algorithm comprises the following steps:

Step-1: User agent requests the task agent to horizontally segment the dataset ac-cording to the total number of segments required.

Step-2: 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.

5. The Parallel Task With Vertical Partitioning (Data-Vp) Algorithm

The second algorithm considered in the exploration of the applicability of Multi Agent Data Mining to parallel Association Rule Mining is the Data Vertical Partitioning (DATA-VP). The DATA-VP algorithm commences by distributing the input dataset over the available number of workers (Data Mining 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 Data Mining agent in terms of a startColNum and endColNum:

$$\text{allocationItemSet} = \{n \mid \text{startColNum} < n \leq \text{endColNum}\}$$

Each Data Mining agent will have its own allocationItemSet which is then used to determine the subset of the input dataset to be considered by the Data Mining agent. Using their allocationItemSet the task agent will partition the data among the workers (DM agents) as follows:

Step-1: Remove all records in the input dataset that do not intersect with the allocationItemSet.

Step-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 sub tree counted by the Data Mining agent.

Step-3: Send the allocated data partition to the corresponding Data Mining agent.

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

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

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

Table 6.1: Dataset

As an example, the ordered data set in Table 6.1 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}, {}, {}, {}, {}}

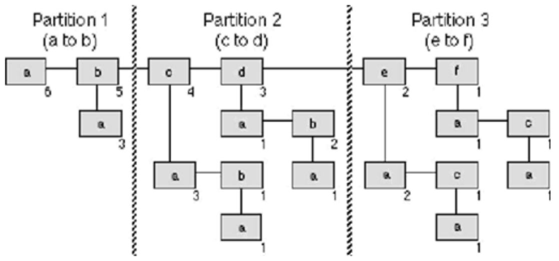


Figure-2: Vertical Partitioning of a T-tree

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-2 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. The DATA-VP Multi Agent Data Mining 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.

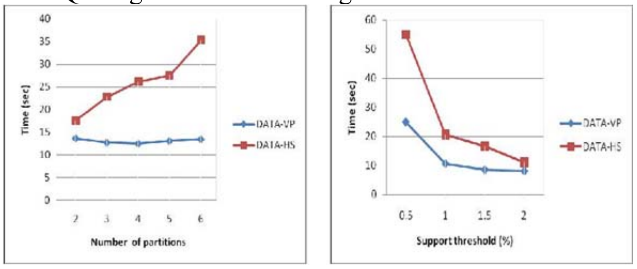
(4) Each worker then generates a T-tree for its allocated partition (a sub T-tree of the final T-tree) using the Apriori-T algorithm as described below.

(5) On completion each Data Mining (worker) agent transmits its partition of the T-tree to the task agent which is then merged into a single global T-tree (the final T-tree ready for the next stage in the Association Rule Mining process, rule generation).

The local T-tree generation process begins with a top-level "tree" comprising only those 1-itemsets included in each worker (Data Mining agent) allocationItemSet. The Data Mining 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 shown in Figure 6.1. 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 Data Mining agents. To avoid the overhead involved in the X-checking process, in this case would require message-passing between the Data Mining agents concerned, X-checking does not take place. Instead, the Data Mining agent will generate its candidates assuming, where necessary, that any subsets outside its local T-tree are frequent.

6. Results And Discussions

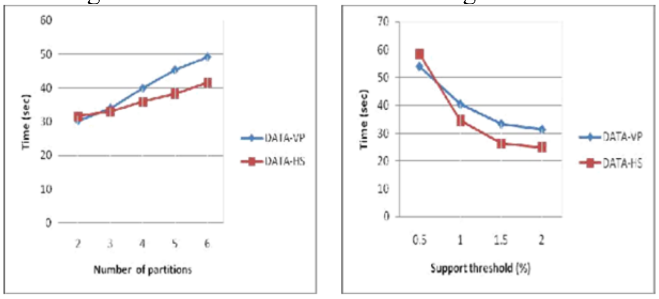
To evaluate the two approaches, in the context of the Extendible Multi Agent Data Mining System vision, a number of experiments were conducted. We consider two artificial datasets: (i) T20.D100K.N250.num, and (ii) T20.D500K.N500.num where T = 20 (average number of items per transactions), D = 100K or D = 500K (Number of transactions), and N = 500 or N = 250 (Number of attributes) are used. The datasets were generated using the IBM Quest generator used in Agrawal and Srikant.



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

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

As noted above the most significant overhead of any parallel system is the number and size of messages sent and received between agents. For the DATA-VP Extendible Multi Agent Data Mining System 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.



(a) Number of Data Partitions (b) Support Threshold
Figure-4: Average of Execution Time for Dataset T20.D500K.N500.num

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 vertical distribution of the input dataset, and (ii) readily lends it to parallelization/distribution. However, when the data size is increased as in the second experiment, and further Data Mining (worker) agents are added (increasing the number of data partitions), the results shown in Figure 6.4, 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.

6.1 Discussion

Multi Agent Data Mining can be viewed as an effective distributed and parallel environment where the constituent agents function autonomously and (occasionally) exchange information with each other. Extendible Multi Agent Data Mining 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 Data Mining can improve both efficiency and scalability first by executing the Data Mining processes in parallel improving the run time efficiency and second, by applying the Data Mining processes on smaller subsets of data that are properly partitioned and distributed to fit in main memory (a data reduction technique). We demonstrated that Multi Agent Data Mining provides suitable mechanisms for exploiting the benefits of parallel computing; particularly parallel data processing. The scenario also demonstrated that Multi Agent Data Mining is suitable for re-usability and illustrated how it is supported by re-employing the Meta Association Rule Mining task agent with the DATA-HS task.

7. Conclusion

Multi Agent Data Mining method for parallel Association Rule Mining has been described so as to explore the Multi Agent Data Mining issues of scalability and re-usability. Scalability is explored by parallel processing of the data and re-usability is explored by re-employing the Meta Association Rule Mining task agent with the DATA-HS task. The solution to the scenario considered in this chapter 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 Association Rule Mining task when employed on Extendible Multi Agent Data Mining System. Each Data Mining 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 T-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 Association Rule Mining when employed on Extendible Multi Agent Data Mining 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. P. O'Brien and R. Nicol. FIPA - towards a standard for software agents, volume 16, no.3, pages (51-59). BT Technology Journal, 1998.
2. R. Bordini, L. Braubach, M. Dastani, A. Seghrouchni, J. Sanz, J. Leite, G. OHare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. In Proceedings of the IEEE International Conference on Cognitive Informatics, pages (33-44), 2006.
3. Reticular Systems. AgentBuilder - An integrated Toolkit for Constructing Intelligence Software Agents. Acronymics, Inc., 1999. Available at http://www.agentbuilder.com.
4. Y. Shoham. Agent-oriented programming, volume 60(1), pages (51-92). In Proceedings of the Artificial Intelligence, Elsevier Science Publishers Ltd. Essex, England, 1993.
5. S. Thomas. The PLACA Agent Programming Language. In Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, pages (355-370), 1994.
6. S. DeLoach and M. Wood. Developing Multiagent Systems with agentTool. In Proceedings of the Intelligent Agents VII. Agent Theories, Architectures, and Languages - 7th International Workshop, ATAL, Boston, MA, USA, 2000.
7. S. DeLoach. Analysis and Design using MASE and agentTool. In Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), 2001.
8. M. Genesereth and S. Ketchpel. Software Agents. In Proceedings of the Communications of the Association for Computer Machinery (ACM), 37(7), pages (48-53), 1994.
9. Munindar P. Singh. Write Asynchronous, Run Synchronous. In Proceedings of the IEEE Internet Computing, 3(2), pages (4-5), 1999.
10 K. Sycara, A. Pannu, M. Williamson, and D. Zeng. Distributed Intelligent Agents, volume 11(6), pages (36-46). IEEE Expert, 1996
11 WEKA. Data Mining Software in Java. The University of Waikato, New Zealand, 1993. http://www.cs.waikato.ac.nz/ ml/weka/.