Scaling Data Mining Algorithms to Large and Distributed Datasets

Shashikumar G. Totad[1], Geeta R. B.[1], Chennupati R Prasanna[2], N Krishna Santhosh[2] , PVGD Prasad Reddy[3]

[1]Department of CSE/IT, GMR Institute of Technology, Rajam, Srikakulam District, AndraPradesh, India. skumartotad@yahoo.com

[2]Department of IT, Vignan's institute of engineering for women, Visakhapatnam, AndraPradesh, India.

[3]Department of CS & SE, Andhra University, Visakhapatnam, AndraPradesh, India.

Abstract. *In the contemporary world of global economy real-life data is distributed and evolving consistently. For the purpose of data mining, the large set of evolving and distributed data can be handled efficiently by Parallel Data mining and Distributed Data Mining, Incremental Data mining. In this paper, we discuss about the issues and the present research work that is being carried out on parallel and distributed data mining. Adaptability of some core data mining algorithms such as decision trees, discovery of frequent patterns, clustering, etc., for parallel processing and contemporary research work related to parallel processing of the algorithms is also discussed. We have identified two approaches for carrying out distributed data mining and tried to bring out the advantages of using mobile agents in client server–based approaches, in terms of bandwidth usage and network latency.*

Keywords: *Classification, Clustering, Parallel data mining, Client-Server, Mobile-Agent.*

1. Introduction

Large databases, some times distributed over several remote locations, are becoming mere common in the contemporary *Global Economy* scenario. The local databases which were initially small, have grown, growing continually and getting distributed to several remote sites as a result of globalization. Many of the conventional data mining algorithms are ineffective and inefficient for handling large and distributed data sets. Hence, the scalability of data mining algorithms has become an active area of research with many challenging problems. There are five basic approaches for scalability of data intensive computing, as identified by G.L Grossman et al. in their work published in [1]. They are:

1. *Manipulating data to fit into memory:* There are four basic variants of this approach, namely Sampling, Feature selection, Partitioning, Data summarization.
2. *Reducing the access time:* This approach tries to reduce the access time for accessing out of memory data by using specialized data structure to access data on disk such as B^+ tree and using some technique such as parallel block reads.
3. *Using multiple processors:* There are two basic techniques used to speed up algorithms: *Task Parallelism & Data Parallelism.*
4. *Pre-computation:* It involves Pre-computation of most expensive part of the algorithm, such as sorting, if possible.

5. *Data Reduction:* Here, along with the techniques used in the above approaches, techniques such as Data Smoothing, Data Compression and Data Transformation can be used.

In the contemporary world of global economy, real-life data is distributed and evolving consistently. The large set of evolving and distributed data can be handled efficiently by *Incremental Data mining, Parallel Data mining and Distributed Data Mining*. In the following sections we discuss about the issues and the present research work that is being carried out on parallel and distributed data mining. In section two parallel data mining is discussed, and in section three about distributed data mining. Each of the above two approaches in turn may use one or more of the approaches discussed earlier, such as Sampling, Pre-computation, data Reduction etc.

2. Scaling Data Mining Algorithms through Parallel Processing

In this section we discuss about some of the basic techniques of parallel processing of large datasets, for data mining tasks and adaptability of some core data mining algorithms (such as decision trees, discovery of frequent patterns, and creating clusters) for parallel processing. Contemporary research work related to parallel processing of the data mining algorithms follows next.

2.1 Basic Techniques of Parallel Processing

The various programming models of parallel processing are characterized by two factors, namely type of parallelism used and type of inter-processor communication used. Data can be divided into partitions, so also the task into subtasks. Accordingly, based on whether a data is divided into partitions or a task into subtasks, three kinds of parallelism can be identified viz., *Data Parallelism, Task Parallelism and Hybrid Parallelism*. Based on how different processors communicate, there are three kinds of parallel processing systems, namely *Shared Memory Systems, Message Passing Systems or Remote Memory Operations System*.

Data Parallelism: In data parallelism the original data set is divided into number of partitions and the same program runs on each of the partitions. Results obtained by running the program on each data partitions is later on combined to get the final result.
Task Parallelism*:* Here, the main task is divided into set of smaller subtasks. The subtasks of this set that can be simultaneously performed are identified and assigned to individual processors. The results obtained by these independent processors are used to get the final solution. This speeds up the execution by multifold as the tasks are performed concurrently.
Hybrid Parallelism*:* In this, both data and tasks are divided. The subtasks that can be performed on independent data partitions, whose results can be combined together, are identified along with the independent subtasks. These subtasks are assigned to independent processors in proper sequence and using the results of these parallel processing, final result is obtained.
Shared Memory Systems: This is the simplest form of communication used in parallel systems. Here, all the processors share a common global memory. Locking of shared memory is used when two or more processors try to use the shared memory for writing.
Message Passing: Here, each processor has its own memory. The processors communicate by sending and receiving the messages explicitly using send and receive commands.
Remote Memory Operations: As in case of message passing, each processor has its own memory. But unlike in message passing, here processors can explicitly access memory of other processors. Separate commands are used for accessing local and remote memory.

2.2 Scope for Parallel Processing in Data Mining Algorithms

In the following section we examine the various contemporary algorithms used at micro and macro level of tasks and discuss the adaptability of these algorithms for parallelism. Most of the data mining tasks involve subtasks such as finding of maximum, minimum, count, average etc, where in they can be performed on independent partitions of a dataset and the sub solutions can be combined to get the final solutions.

Parallelism in Association Rule Mining: The Apriori algorithm [2] is used for finding frequent patterns. In Apriori algorithm, set of frequent patterns of size k+1 is obtained in two steps. The first step finds a set of candidate frequent item sets through join operation of set of frequent items of size K. In the second step, the entire dataset is scanned to count the number of times the candidate frequent patterns appear together in all the transactions. As discussed above the second step can be performed by distributing the dataset to several processors simultaneously.

Concurrency can also be used in the first step to speed up the creation of candidate frequent-pattern set, by distributing the candidate frequent patterns themselves among the processors. Further, based upon the observation by Cheung at el. [3] that a global frequent item set must be a local frequent item for some data partition, local pruning can be applied by removing the sets that or not locally frequent. This also reduces the communication required for exchanging support counts.

The FP–growth [4] algorithm for finding frequent patterns can be on independent partitions of the dataset to construct one tree for each data partition simultaneously. The individual FP-trees can be combined to have the single tree that would be constructed by applying FP-growth on the entire undivided dataset.

Parallelism in Classification: *Decision Tree Induction* algorithm [6] is used for classification by constructing a decision tree. The algorithm constructs decision tree recursively using depth–first divide and conquer approach. At any given node, to further split up the dataset towards identification of a class, the algorithm chooses the most suitable attribute based on the *information gain* value of the attributes. The information gain of an attribute is a measure of the ability of an attribute to minimize the information needed to classify the given entity in the resulting sub-trees. Either task parallelism or data parallelism can be used for construction of decision tree.

For task parallelism, each processor simultaneously computes information gain based on one attribute of the remaining attributes. The processors exchange the each others information gain and decide for the splitting attribute. For data parallelism, the entire data set is divided into partitions and one partition is distributed to each processor. Each processor builds a separate tree for its data partition. The ensemble of trees obtained by different processors is combined to obtain the final decision tree.

Parallelism in Clustering: Clustering is considered to be unsupervised learning for classification where in the number of classes into which the data is to be grouped is not given or the data does not contain any attribute of class information. Parallelism in clustering can be exploited by trying clustering algorithm on different cluster numbers simultaneously, one by each processor. Also, the distance measure is the mostly used metric for computing inter–cluster dissimilarity and intra–cluster similarity. One processor each can be used for simultaneously calculating the distances of one cluster center from all other data points.

2.3Existing Parallel Algorithms:

In the following sections we elaborate on the present research work on parallelism of data mining tasks such as Frequent-Pattern mining, Classification, and Clustering on large dataset.

Existing Frequent-Pattern Mining Algorithms: All the parallel algorithms proposed for mining frequent patterns are either Apriori–based techniques or FP–tree–based techniques. Agrawal and Shafer proposed three different strategies for parallelizing Aprori algorithm in [7]. Three strategies are based on count distribution, data distribution, and candidate FP–distribution. In their work the authors have discussed about advantages and limitations of the three strategies with respect to computational costs, communication overheads and memory usage. Orlando et al. [8] and Ashrafi et al. [9] extended the work done by Agrawal & Shafer by further improving the speed and efficiency of the approaches proposed in [7]. A. Salvasere et al. [10] proposed a partition–based parallel algorithm that performs efficient partitioning of dataset. All of above proposed algorithms are Apriori–based and they suffer from limitations inherent to Apriori (such as multiple scans).

Most of the FP–tree–based parallel algorithms proposed are based on FP–growth [4] algorithm. Most of these algorithms exploit the parallelism by partitioning the dataset into several parts and then build local FP–tree for each part in parallel [11], [12], [13], where as some algorithms build a global FP-tree for entire database by parallelizing the tree – construction process, such as in [14], [15]. However, the major drawback of these approaches is the excessive communication overhead for broadcasting the itemset–based prefix–trees to all processors, multiple times. The approach in [15] is to build only one FP–tree and to partition this tree itself into several independent parts to mine frequent patterns in parallel by assigning each part to one processor. However, this approach too has the inter-processor communication cost required for communicating parts of the global tree (apart form memory constraint). All of the above algorithms, both Apriori-based and FP–tree–based, require at least two database scans.

Syed K. Tanbeer et al. [16] proposed an approach for parallel mining of frequent patterns that scans the entire data base only once. In their approach, the algorithm uses a novel tree structure called PP–tree (parallel pattern tree) that significantly reduces the I/O cost by capturing the database contents with a single scan. It also reduces inter-processor communication overhead by running independently at each processor and merging the locally generated global frequent patterns at the final stage.

Existing Parallel Algorithms for Classification: Classification is considered to be one of the major data mining tasks. There exist different modules of classification such as neural networks, decision trees, genetic algorithms, and statistical models. Decision trees are relatively faster to build and sometimes give better accuracy compared to other classification methods. Parallel implementation of decision tree appears to be more natural and easy but in fact is more complex for the following two reasons:

- Shape of the decision tree is highly irregular and the amount of processing required at each node may vary significantly. Hence, any static allocation scheme will probably suffer from high load imbalance problem.
- When the children of a node are processed in parallel, their construction requires sharing part of the data of their parent node and hence requires dynamic data movement between processors.

There are three most popular implementations of decision tree, namely SLIQ [18], SPRINT [19], and C4.5 [17]. These algorithms find the best split attribute as the attribute with the highest rank. Ranks of the attribute are calculated using statistical measures such as information gain. To find a best split attribute of continuous valued attributes, SLIQ maintains separate list of stored values for each attribute of the given training dataset and a separate memory-resident class list values of the class attribute, as shown in Fig.1, for the given dataset of Table-1. SPRINT eliminates the class list by adding class list attribute values for each attribute list. The index in the attribute list is now index of the data record in the training dataset, as shown in Fig.2 that considers the sample dataset of Table 1.

Table 1. Sample dataset

TID	Attendance %	Course	Grade
1	90	DS	A
2	75	CC	B
3	65	SE	C
4	85	FLAT	A
5	70	MP	B

The training data set is horizontally and evenly distributed among the parallel processors. Each processor evaluates possible splits by computing ranks of the attributes. The possible splits are communicated between the processors to find the best split. The main drawback of SLIQ and SPRINT is the high inter-processor communication involved in the operation to perform the split of the dataset associated with node.

Attendance %	TID
90	1
85	4
75	2
70	5
65	3

Course	TID
CC	2
DS	1
FLAT	4
MP	5
SE	3

Disk-resident Attribute Lists

TID	Grade
1	A
2	B
3	C
4	A
5	B

Memory-resident Class-list

Fig.1 Data structures used in *SLIQ*.

Attendance %	Grade	TID
90	A	1
75	B	2
65	C	3
85	A	4
70	B	5

Course	Grade	TID
DS	A	1
CC	B	2
SE	C	3
FLAT	A	4
MP	B	5

Fig. 2 Attribute *List* data Structure used in *SPRINT* for each attribute

In paper [17], new parallel implementation of C4.5 decision tree is proposed that uses hybrid parallelism strategy. Data parallelism is used at the beginning of the decision tree construction process, which is similar to one used by SLIQ. It achieves uniform load balance by distributing data records horizontally and equally among the processors and uses breadth-first strategy to build decision tree. Task parallelism is used at the lower nodes of the tree that cover a smaller number of data records. The algorithm proposed in [17] also deals with missing values of attributes and reduces the communication overhead significantly by employing task parallelism for building the nodes of decision tree at lower levels, covering a small portion of the entire dataset.

Existing Parallel Algorithms for Clustering: There have been very few significant research works towards exploration of parallelism in clustering algorithms. In 1995, Olson in his paper [20] suggested a parallel algorithm for hierarchical clustering that uses several distance metrics in parallel. In 1998, Subramanian [21] suggested parallel algorithm for density-based EM (Expectation Maximization) algorithm. He exploited the inherent parallelism of the algorithm by combining task-parallelism and data-parallelism. He exploited the data parallelism by computing cluster model in parallel, given the cluster number. The task-parallelism is exploited by concurrently searching cluster numbers using parallel machine clusters.

3. Distributed Data Mining

The rapid development of computers together with communication networks led to the generation of huge datasets by software applications developed for various services such as supermarket, banking, stock market and mobile service. Distributed Data Mining (DDM) can be implemented using one of the four approaches, mentioned below [22].

- Move the disperse data to a central site, and then perform data mining on the entire data. This involves heavy communication cost to pool the entire data to a central site. Another problem with this approach is the preservation of data privacy which can't be guaranteed.
- Second approach is to perform mining locally at each site to build a local model. All the local models built so are moved to central site to combine them in to a global model. This approach is used in the works of [23] [24] [25]. This approach suffers from scalability problem with increase in number of sites.
- The third approach is using samples. A small set of representative data is carefully sampled at each site to form one global representative dataset. Data mining can then be performed on the global representative data set.
- Unlike all the previous three approaches where a central site to facilitate distributed data mining is involved, here in the fourth approach data mining is performed on P2P networks, where sites communicate directly with each other without involving a central server [26][27].

Based on the use of underlying network infrastructure, distributed data mining systems can be broadly categorized into Client-Server DDM systems, and Mobile agent-based DDM systems. In the following sections we discuss about some of the research works done under these.

3.1 Client-server DDM Systems

In client server model data from different sites are collected at data mining server as shown in Fig.3. The data mining server is normally a high performance server and performs data mining on the collected data for the mining request from clients. The advantage of the client-server model is that the server has high performance computational resources for handling resource intensive data mining tasks. However, this model involves high communication overhead as the distributed data has to be collected at central server.

Two significant DDM models that use client-server system are proposed in [29] and [30]. Parthsarthy .S et al. [30] proposed a DDM system called *Intelliminer* that implements distributed *doall* (D-DOALL) primitive by proposing two different "resource-aware" scheduling algorithms. A *doall* loop is one in which the loop iterations are independent, i.e. where there are no conflicts between iterations.
EXAMPLE: for (i=0; i<N; i++) { A[i]=B[i] }

Decision center is another major client–server based DDM system proposed by Chattratichat J. et al. [29]. In their work the authors have discussed implications of the requirements of large organizations for accessing and combining of data to provide comprehensive analysis. They suggested use of existing networking infrastructure, typically based on the internet technology.

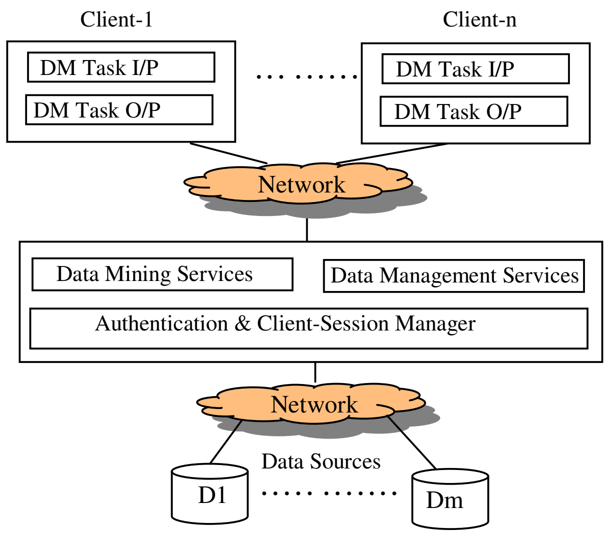

Fig. 3: Client-Server Model of Distributed Data Mining

3.2 Mobile-Agent-Based DDM Systems

A mobile agent is a composition of both code and data which suspends execution and migrates from one computer on the network of computers to another autonomously and continues its execution on the destination computer. The paper [31] discusses the advantages of mobile agents over client server–based approaches by analyzing band with usage and network latency.

Mobile-agent based DDM systems reduce the high communication overhead involved in client-server based DDM systems. Here, when a client submits a data mining query, mobile-agents that perform data mining task are dispatched to the data servers containing the datasets, as shown in the Fig.4. The mobile agents after performing the data mining task on the datasets return with mining results. Using mobile agents has become a natural choice for many of the distributed data mining applications as it facilitates movement of small sized mobile agents that carry code rather than movement of huge volumes of data. Several distributed data mining systems that use mobile agents have been proposed by researchers in the recent past [28] [32] [33].

The work proposed in [28] finds frequent-patterns using mobile-agents by computing *Local Frequent Itemsets* and G*lobal Frequent Itemsets*/*Candidate Global Frequent Itemsets* simultaneously. Also, the computation of *Candidate Global Frequent Itemsets* is performed in overlapped fashion. The work demonstrates that the time taken for computing *Global Frequent Itemsets* is reduced significantly by using mobile agents. The work published in paper [32] demonstrates how set of mobile agents having specialized expertise can be used to gather relevant information and how they cooperate with each other to arrive at timely decisions, in dealing with various supply chain finance-business analysis scenarios. The work in [33] proposes a structure for service–oriented software to construct distributed data mining system in e-commerce, using web services technology and mobile-agent technology. The structure proposed is open and can effectively integrate a variety of mining tools. It can even operate the data in heterogeneous environment.

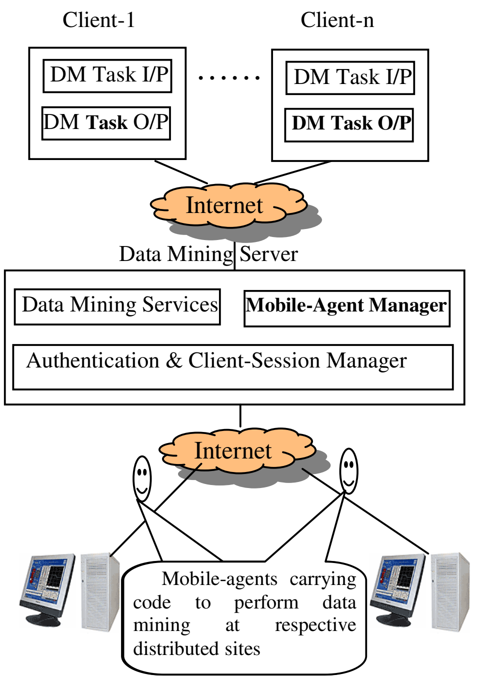

Fig. 6 Mobile-Agent based model of Distributed Data mining System

4. Conclusion

Due to the rapid advancement in information technology, web services in particular, a lot of data is being generated continuously like a stream called "*data stream*". A data stream can be considered as an ordered sequence of transactions that is generated in timely order. Performing data mining for Classification, Clustering, and Frequent pattern discovery in particular in data streams is more challenging for the following reasons:

- By nature, data streams are continuous. Hence, the data streams have to be read and analyzed online and are not available again for multiple data scans.
- Data in streams is seldom uniformly distributed. That is, currently frequent pattern may become infrequent in future and vice versa. Hence, infrequent patterns are also to be maintained without pruning.

Metwally et al. [5] has proposed a new algorithm for discovering frequent patterns in data streams. Some existing techniques can be adopted for finding frequent patterns in data streams. In the present scenario of web services and ever evolving distributed databases it is highly essential to scale up the existing data mining algorithms or to design new data mining algorithms. Parallel and distributed processing can be used and, in our opinion, is natural option for solving the scalability problem of the large databases. In our opinion, exploration of the possibility of applying suitable form of parallel processing to some of the existing techniques, to enhance computational speed, has also a great research scope.

References

[1]. R. L. Grossman and Yike Guo, Parallel Methods for Scaling Data Mining Algorithms to Large Datasets, Handbook on Data Mining and Knowledge Discovery, Oxford University Press, 2002 pages 433-442.

[2]. Agrawal R, Srikant R. “Fast Algorithms for Mining Association Rules”. In Proc. of *VLDB,* Sep 2-15 1994, pp. 487-99.

[3]. D W Cheung, J. Han, V.T. Ng, and C.Y. Wong. ”Maintenance of discovered association rules in large databases: an incremental updating technique”. In Proc. of *ICDE 1996*, pp. 106–114.

[4]. J. Han, J. Pei, Y. Yin and R. Mao, “Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach”. Data Mining and Knowledge Discovery, 8(1), 2004, pp.53-87.

[5]. Metwally A, Agrawal D, & Abbadi A. E. “Efficient Computation of Frequent and Top-k Elements in Data streams”. International Conference on Data Theory, 2005, pp. 398-412.

[6]. R. Quinlan, “Induction of Decision Trees”, Machine Learning,1, 1986, pp. 81–106.

[7]. Agrawal R. and Shafer, J.C. “Parallel mining of association rules”. IEEE Transactions on Knowledge and. Data Engineering, 1996, Vol. 8, No.6, pp. 962-969.

[8]. S. Orlando, P. Palmerini, R. Perego and F. Silvestri. "An Efficient Parallel and Distributed Algorithm for Counting Frequent Sets". Proceedings of Int. Conf. VECPAR 2002, June 2002.

[9]. M. Z. Ashrafi, D. Taniar and K. Smith. “ODAM: An Optimal Distributed Association Rule Mining Algorithm”.IEEE Distributed Systems Online,5(3), 2004, pp.1541-4922.

[10]. A. Savasere, E. Omiecinski, and S. B. Navathe. “An Efficient Algorithm for Mining Association Rules in Large Databases”. VLDB, 1995, pp. 432-444.

[11]. O. R. Zaiane,M. El-Hajji, and P. Lu. “Fast Parallel Association Rule Mining without candidacy Generation”. IEEE International Conference on Data Mining, 2001, pp. 665-668.

[12]. A. Javed and A. Khokhar. “Frequent Pattern Mining on Message Passing Multiprocessor Systems”. Distributed and Parallel Databases, 2004, Vol. 16, pp. 321-324.

[13]. K. M. Yu, J. Jhou, J. Holfinger, and W. C. Hsiao. “Load Balancing Approach Parallel Algorithm for Frequent Pattern Mining”. PaCT, 2007, pp. 623-631.

[14]. G. Buehrer , S. Parthsarathy, S. Tatikonda, T. Kurc, and J. Saltz. “Towards Terabyte Pattern Mining an Architecture conscious Solution”. PPoPP, 2007, pp. 2-12.

[15]. [22] D. Chen, C. Lai, W. Hu, W. G. Chen, Y. Zhang, and W. Zheng.“ The Partition based Parallel Frequent Pattern Mining on Shared Memory Systems“. IEEE Parallel and Distributed Processing Symposium, 2006.

[16]. Syed K. Tanbeer, C. F. Ahmed, B-S Jeong . “Parallel and Distributed Algorithms for Frequent Pattern Mining in Large Databases”. IETE Technical Review, 2009, Vol.26, Issue 1.

[17]. N. Amado, J. Gama, F. Silva. “Exploiting Parallelism in Decision Tree Induction”. 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, 2003.

[18]. M. Mehta, R. Agrawal, J. Rissanen . “SLIQ: A Fast Scalable Classifier for Data Mining”, 5th International Conference on Extending Database Technology: Advances in Database Technology, 1996, pp. 18-32.

[19]. J. C. Shafer, R. Agrawal, and M. Mehta. “SPRINT: A Scalable Parallel Classifier for Data Mining”. In Proc. 22nd Int. Conf. on Very Large Databases, 1996, pp. 544–555.

[20]. Olson, C.F. "Parallel Algorithms for Hierarchical Clustering.". In *Parallel Computing*, 1995, 21(8), pp.1313-1325.

[21]. Subramonian, R. and Parthasarathy, S. “A Framework for Distributed Data Mining". In Proceedings of KDD98 Workshop on Distributed Data Mining, 1998.

[22]. Khaled M. Hammouda and Mohamed S. Kamel. “Hierarchically Distributed Peer-to-Peer Document Clustering and Cluster Summarization”. IEEE Transactions on Knowledge and Data Engineering, , 2009, Vol. 21(5), pp.681-698.

[23]. N.F. Samatova, G. Ostrouchov, A. Geist, and A.V. Melechko. “RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets”. Distributed and Parallel Databases, 2002,vol. 11(2), pp. 157-180.

[24]. S. Merugu and J. Ghosh, “Privacy-Preserving Distributed Clustering Using Generative Models” . 3rd IEEE Int’l Conf. Data Mining (ICDM ’03), 2003, pp. 211-218.

[25]. J. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch, “Distributed Data Mining and Agents,” Eng. Applications of Artificial Intelligence, 2005,vol.18(7) , pp. 791-807.

[26]. S. Datta, C. Giannella, and H. Kargupta. "K-Means Clustering over Peer-to-Peer Networks". 8th Int. Workshop on High Performance and Distributed Mining (HPDM), 2005.

[27]. S. Datta, C. Giannella, and H. Kargupta, "K-Means Clustering over a Large, Dynamic Network," 6th SIAM Int. Conf. Data Mining (SDM '06),2006, pp. 153-164.

[28]. U.P.Kulkarni , P.D.Desai ,Tanveer Ahmed , J.V.Vadavi , A.R.Yardi . "Mobile Agent Based Distributed Data Mining". International Conference on Computational Intelligence and Multimedia Applications, 2007, pp. 18-24.

[29]. Chattratichat,J., Darlington, J , et al, "An Architecture for Distributed Enterprise Data Mining", 7th Intl. Conf. on High Performance Computing and Networking, 1999.

[30]. Parthsarthy S., and Subramanian R. " Facilitating Data mining on a Network of Works Stations", AAAI Press, 1999.

[31]. U.P.Kulkarni, A.R.Yardi, et al, "Exploring the Capabilities of Mobile Agents in Distributed ata Mining", 10th Intl. Database Engineering & Applications Symposium- IDEA'06, IEEE-06.

[32]. LIU Xiang. "An Agent-based Architecture for Supply Chain Finance Cooperative Context-aware Distributed Data Mining Systems". 3rd Intl. Conference on Internet and Web Applications and Services, IEEE-2008.

[33]. Jinbiao Hou. "Research on Distributed Data Mining in E-commerce Environment Based on Web Services and Mobile Agent". Intl. Conf. on Computational Intelligence and Natural Computing, IEEE-2009.