Parallelism in Knowledge Discovery Techniques

Domenico Talia

DEIS, Università della Calabria, Via P. Bucci, 41c
87036 Rende, Italy
talia@deis.unical.it

Abstract. Knowledge discovery in databases or data mining is the semi-automated analysis of large volumes of data, looking for the relationships and knowledge that are implicit in large volumes of data and are 'interesting' in the sense of impacting an organization's practice. Data mining and knowledge discovery on large amounts of data can benefit of the use of parallel computers both to improve performance and quality of data selection. This paper presents and discusses different forms of parallelism that can be exploited in data mining techniques and algorithms. For the main data mining techniques, such as rule induction, clustering algorithms, decision trees, genetic algorithms, and neural networks, the possible ways to exploit parallelism are presented and discussed in detail. Finally, some promising research directions in the parallel data mining research area are outlined.

1 Introduction

Today the information overload is a problem like the shortage of information. In our daily activities we often deal with flows of data much more larger than we can understand and use. Thus we need a way to sift those data to extract what is interesting and relevant for our activities. Knowledge discovery in databases, also called data mining, is the semi-automated analysis of large volumes of data, looking for the relationships and knowledge that are implicit in large volumes of data and are 'interesting' in the sense of impacting an organization's practice. Research and development work in the area of knowledge discovery and data mining concerns the study and definition of techniques, methods, and tools for the extraction of novel, useful, and implicit patterns from data.

Knowledge discovery in large data repositories can find what is interesting in them representing it in an understandable way [3]. Mining large data sets requires large computational resources. In fact, data mining algorithms working on very large data sets take very long times on conventional computers to get results. One approach to reduce response time is sampling. But, in some case reducing data might result in inaccurate models, in some other case is not useful (e.g., outliers identification). The other approach is parallel computing. High performance computers and parallel data mining algorithms can offer a very efficient way to mine very large data sets [8] [17] by analyzing them in parallel.

Is not uncommon to have sequential data mining applications that require several days or weeks to complete their task. Parallel computing systems can

J. Fagerholm et al. (Eds.): PARA 2002, LNCS 2367, pp. 127–136, 2002.

bring significant benefits in the implementation of data mining and knowledge discovery applications by means of the exploitation of inherent parallelism of data mining algorithms. Data mining and knowledge discovery on large amounts of data can benefit of the use of parallel computers both to improve performance and quality of data selection. When data mining tools are implemented on high-performance parallel computers, they can analyze massive databases in a reasonable time. Faster processing also means that users can experiment with more models to understand complex data. Furthermore, high performance makes it practical for users to analyze greater quantities of data.

This paper presents and discusses different forms of parallelism that can be exploited in data mining techniques and algorithms. The main goal of the paper is to introduce data mining techniques on parallel architectures and show how large scale data mining and knowledge discovery applications can be scalable by using systems, tools and performance offered by parallel processing systems. For several data mining techniques, such as rule induction, clustering algorithms, decision trees, genetic algorithms, and neural networks, different strategies to exploit parallelism are presented and discussed. Furthermore, some experiences and results in parallelizing data mining algorithms according to different approaches are discussed. Finally, some promising research directions in parallel data mining are outlined.

2 Data Mining and Parallel Computing

Main goals of the use of parallel computing technologies in the data mining field are:

- performance improvements of existing techniques,
- implementation of new (parallel) techniques and algorithms, and
- concurrent analysis using different data mining techniques in parallel and result integration to get a better model (that is more accurate).

We identify three main strategies in the exploitation of parallelism in data mining algorithms:

1. *independent parallelism*,
2. *task parallelism*,
3. *SPMD parallelism*.

Independent parallelism is exploited when processes are executed in parallel in an independent way; generally each process has access to the whole data set and does not communicate or synchronize with other processes. According to task parallelism (or control parallelism) each process executes different operations on (a different partition of) the data set. Finally, in Single Program Multiple Data (SPMD) parallelism a set of processes execute in parallel the same algorithm on different partitions of a data set and processes cooperate to exchange partial results. These three strategies are not necessarily alternative for parallelizing data mining algorithms. They can be combined to improve

both performance and accuracy of results. In combination with strategies for parallelization, different data partition strategies can be used :

a. *sequential partitioning*: separate partitions are defined without overlapping among them;
b. *cover-based partitioning*: some data can be replicated on different partitions;
c. *range-based query partitioning*: partitions are defined on the basis of some queries that select data according to attribute values.

3 Parallelism in Data Mining Techniques

This section presents different parallelization strategies for each data mining technique and outlines some parallel data mining tools, algorithms or systems.

Table 1 contains the main data mining tasks and for each task the main techniques used to solve them are listed. In the following sections we describe different approaches for parallel implementation of some techniques listed in table 1.

Table 1. Data mining tasks and used techniques

Data Mining Tasks	**Data Mining Techniques**
Classification	induction, neural networks, genetic algorithms
Association	Apriori, statistics, genetic algorithms
Clustering	neural networks, induction, statistics
Regression	induction, neural networks, statistics
Episode discovery	induction, neural networks, genetic algorithms
Summarization	induction, statistics

3.1 Parallel Decision Trees (Parallel Induction)

Classification is the process of assigning new objects to predefined categories or classes. Decision trees are an effective technique for classification. They are tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a data set. The tree leaves represents the classes and the tree nodes represents attribute values. The path from the root to a leaf gives the features of a class in terms of attributes-values couples.

Task parallel approach. According to the task parallelism approach one process is associated to each sub-tree of the decision tree that is built to represent a classification model. The search occurs in parallel in each sub-tree, thus the degree of parallelism P is equal to the number of active processes at a given time. A possible implementation of this approach is based on farm parallelism in which there is a master process that controls the computation and a set of P workers that are assigned to the sub-trees.

SPMD approach. In the exploitation of SPDM parallelism each process classifies the items of a subset of data. The P processes search in parallel in the

whole tree using a partition D/P of the data set D. The global result is obtained by exchanging partial results. The data set partitioning can be operated in two main different ways:

- by partitioning the D tuples of the data set: D/P per processor.
- by partitioning the n attributes of each tuple: D tuples of n/P attributes per processor.

In [11] a parallel implementation of the C4.5 algorithm that use the independent parallelism approach is discussed. Other significant examples of parallel algorithms that use decision trees are SPRINT discussed in [16], and TDIDT (Top-Down Induction of Decision Trees) [15].

3.2 Discovery of Association Rules in Parallel

Association rule algorithms, such as Apriori, allow automatic discovery of complex associations in a data set. The task is to find all frequent itemsets, i.e. to list all combinations of items that are found in a sufficient number of examples. Given a set of transactions D, the problem of mining association rules is to generate all association rules that have support (how often a combination occurred overall) and confidence (how often the association rule holds true in the data set) greater than the user-specified minimum support and minimum confidence respectively. An example of such a rule might be that "*98% of customers that purchase tires and auto accessories also get automotive services done*".

SPMD approach. In the SPMD strategy the data set D is partitioned among the P processors but candidate itemsets I are replicated on each processor. Each process p counts in parallel the partial support S_p of the global itemsets on its local partition of the data set of size D/P. At the end of this phase the global support S is obtained by collecting all local supports S_p. The replication of the candidate itemsets minimizes communication, but do not use memory efficiently. Due to low communication overhead, scalability is good.

Task parallel approach. In this case both the data set D and the candidate itemsets I are partitioned on each processor. Each process p counts the global support S_i of its candidate itemset I_p on the entire data set D. After scanning its local data set partition D/P, a process must scan all remote partitions for each iteration. The partitioning of data set and candidate itemsets minimizes the use of memory but requires high communication overhead in distributed memory architectures. Due to communication overhead this approach is not scalable as the previous one.

Hybrid approaches. Combination of different parallelism approaches can be designed. For example, SPMD and task parallelism can be combined by defining C clusters of processors composed of the same number of processing nodes. The data set is partitioned among the C clusters, thus each cluster is responsible to compute the partial support S_c of the candidate itemsets I according to the SPMD approach. Each processor in a cluster uses the task parallel approach to compute the support of its disjoint set of candidates I_p by scanning the data set

stored on the processors of its cluster. At the end of each iteration the clusters cooperate each other to compute the global support S.

The Apriori algorithm [1] is the most known algorithm for association rules discovery. Several parallel implementations have been proposed for this algorithm. In [2] are presented two different parallel algorithms called Count Distribution (CD) and Data Distribution (DD). The first one is based on independent parallelism and the second one is based on task parallelism. In [9] are presented two different parallel approaches to Apriori called Intelligent Data Distribution (IDD) and Hybrid Distribution (HD). A complete review of parallel algorithms for association rules can be found in [18].

3.3 Parallel Neural Networks

Neural networks (NN) are a biology-inspired model of parallel computing that can be used in knowledge discovery. Supervised NN are used to implement classification algorithms and unsupervised NN are used to implement clustering algorithms. A lot of work on parallel implementation of neural networks has been done in the past. Theoretically, each neuron can be executed in parallel, but in practice the grain of processors is generally larger of grain of neurons. Moreover, the processor interconnection degree is restricted in comparison with neuron interconnection. Hence a subset of neurons is generally mapped on each processor. There are several different ways to exploit parallelism in a neural network:

1. *parallelism among training sessions*: it is based on simultaneous execution of different training sessions,
2. *parallelism among training examples*: each processor trains the same network on a subset of $1/P$ examples,
3. *layer parallelism*: each layer of a neural network is mapped on a different processor,
4. *column parallelism*: the neurons that belong to a column are executed on a different processor,
5. *weight parallelism*: weight summation for connections of each neuron is executed in parallel.

These parallel approaches can be combined to form different hybrid parallelization strategies. Different combinations can raise different issues to be faced for efficient implementation such as interconnection topology, mapping strategies, load balancing among the processors, and communication latency.

Typical parallelism approaches that are used for the implementation of neural networks on parallel architectures are task parallelism, SPMD parallelism, and farm parallelism.

Clementine is a parallel data mining system based on neural nets. Several task-parallel implementations of back-propagation networks parallel implementations of a Self-organizing maps have been implemented for data mining tasks. Finally, Neural Network Utility (NNU) [4] is neural network-based data mining environment that has been also implemented on a IBM SP2 parallel machine.

3.4 Parallel Genetic Algorithms

Genetic algorithms are used today for several data mining tasks such as classification, association rules, and episode discovery. Parallelism can be exploited in three main phases of a genetic algorithm:

- population initialization,
- fitness computation, and
- execution of the mutation operator,

without modifying the behavior of the algorithm in comparison to the sequential version. On the other hand, the parallel execution of selection and crossover operations requires the definition of new strategies that modify the behavior (and results) of a genetic algorithm in comparison to the sequential version. The most used approach is called *global parallelization*. It is based on the parallel execution of the fitness function and mutation operator while the other operations are executed sequentially. However, the are two possible SMPD variants:

a. Each processor receives a subset of elements and evaluates their fitness using the entire data set D.
b. Each processor receives a subset D/P of the data set and evaluates the fitness of every population element (data item) on its local subset.

Global parallelization can be effective when very large data sets are to be mined. This approach is simple and has the same behavior of its sequential version, however its implementations did not achieve very good performance and scalability on distributed memory machines because of communication overhead.

Two different parallelization strategies that can change the behavior of the genetic algorithm are the *island model* (coarse grained) where each processor executes the genetic algorithm on a subset N/P of elements (sub-demes) and periodically the best elements of a sub-population are migrated towards the other processors, and the *diffusion model* (fine grained) where population is divided in a large number of sub-populations composed of few individuals (D/n where $n >> P$) that evolve in parallel. Several subsets are mapped on one processor. Typically, elements are arranged in a regular topology (e.g., a grid). Each element evolves in parallel and executes the selection and crossover operations with the neighbor elements.

A very simple strategy is the independent parallel execution of P independent copies of a genetic algorithm on P processors. The final result is selected as the best one among the P results. Different parameters and initial populations should be used for each copy. In this approach there is no communication overhead. The main goal here is not getting a higher performance but a better accuracy. Some significant examples of data mining systems based on the parallel execution of genetic algorithms are GA-MINER, REGAL [13], and G-NET.

3.5 Parallel Cluster Analysis

Clustering algorithms arrange data items into several groups, called *clusters* so that similar items fall into the same group. This is done without any suggestion

from an external supervisor, so classes are not given a priori but they must be discovered by the algorithm. When used to classify large data sets, clustering algorithms are very computing demanding.

Clustering algorithms can roughly be classified into two groups: hierarchical and partitioning models. Hierarchical methods generate a hierarchical decomposition of a set of N items represented by a *dendogram*. Each level of a dendogram identifies a possible set of clusters. Dendograms can be built starting from one cluster and iteratively this cluster is split until N clusters are obtained (*divisive methods*) or starting with the N clusters and at each step two clusters are merged until only one is left (*agglomerative methods*).

Partitioning methods divide a set of objects into K clusters using a distance measure. Most of these approaches assume that the number K of groups has been given a priori. Usually these methods generate clusters by optimizing a criterion function. The K-means clustering is a well-known and effective method for many practical applications that employs the squared error criterion.

Parallelism in clustering algorithms can be exploited both in the clustering strategy and in the computation of the similarity or distance among the data items, by computing on each processor the distance/similarity of a different partition of items. In the parallel implementation of clustering algorithms the three main parallel strategies described in section 2 can be exploited.

Independent parallel approach. Each processor uses the whole data set D and it performs a different classification based on a different number of clusters K_p. To get the load among the processors balanced, until the clustering task is complete a new classification is assigned to a processor that completed its assigned classification.

Task parallel approach. Each processor executes a different task that composes the clustering algorithm and cooperates with other processors exchanging partial results. For example, in partitioning methods processors can work on disjoint regions of the search space using the whole data set. In hierarchical methods a processor can be responsible of one or more clusters. It finds the nearest neighbor cluster by computing the distance among its cluster and the others. Then all the local shortest distances are exchanged to find the global shortest distance between two clusters that must be merged. The new cluster will be assigned to one of the two processors that handled the merged clusters.

SPMD approach. Each processor executes the same algorithm on a different partition D/P of the data set to compute partial clustering results. Local results are then exchanged among all the processors to get global values on every processor. The global values are used in all processors to start the next clustering step until a convergence is reached or a given number of steps are executed. The SPMD strategy can be also used to implement clustering algorithms where each processor generates a local approximation of a model (classification) that at each iteration can be passed to the other processors that can use it to improve their clustering model.

In [14] it can be found a set of hierarchical clustering algorithms and an analysis of time complexity on different parallel architectures. An example of

parallel implementation of a clustering algorithm is P-CLUSTER [10]. Other parallel clustering algorithms are discussed in [5], [12], and [7]. In particular, in [7] an SPDM implementation of the AutoClass algorithm, named P-AutoClass is described. The paper shows interesting performance results on distributed memory MIMD machines. Table 2 shows experimental performance results we obtained by running P-AutoClass on a parallel machine using up to 10 processors for clustering a data set composed of 100,000 tuples with two real valued attributes. In particular, table 2 contains execution times (in secs) and absolute speedup on 2, 4, 6, 8 and 10 processors. We can observe as the system behavior is scalable; speedup on 10 processors is about 8 and execution time significantly decreases from 245 to 31 minutes.

Table 2. Execution time and speedup of P-AutoClass

Processors	Execution Time (secs)	Speedup
1	14683	1.0
2	7372	2.0
4	3598	4.1
6	2528	5.8
8	2248	6.5
10	1865	7.9

4 Architectural Issues

In presenting the different strategies for the parallel implementation of data mining techniques we not addressed architectural issues such as

- distributed memory versus shared memory implementation,
- interconnection topology of processors,
- optimal communication strategies,
- load balancing of parallel data mining algorithms,
- memory usage and optimization, and
- I/O impact on algorithm performance.

These issues (and others) must be taken into account in the parallel implementation of data mining techniques. The architectural issues are strongly related to the parallelization strategies and there is a mutual influence between the knowledge extraction strategy and the architectural features. For instance, increasing the parallelism degree in some case corresponds to an increase of the communication overhead among the processors. However, communication costs can be also balanced by the improved knowledge that a data mining algorithm can get from parallelization. At each iteration the processors share the approximated models produced by each one of them. Thus each processor executes a next iteration using its own previous work and also the knowledge produced by the other processors. This approach can improve the rate at which a data mining algorithm finds a model for data (knowledge) and make up for lost time in communication.

5 Research Issues and Directions

Parallel execution of different data mining algorithms and techniques can be integrated to obtain a better model not just to get high performance but also high accuracy. Here we list some promising research issues in the parallel data mining area:

- It is necessary to develop environments and tools for interactive high performance data mining and knowledge discovery;
- The use of parallel knowledge discovery techniques in text mining must be extensively investigated;
- Parallel and distributed Web mining is a very promising area for exploiting high-performance computing techniques;
- The integration of parallel data mining techniques with parallel data warehouses is a crucial aspect for private enterprizes and public organizations.

Besides these very promising area we would like to mention the importance of the integrated use of clusters and grids for distributed and parallel knowledge discovery. Grid integrated clusters of computers that execute the same or different data mining or KDD algorithms can be seen as a massively parallel computers that mine very large data sets. The development of software architectures, environments and tools for grid-based data mining will result in Grid-aware parallel and distributed knowledge discovery (PDKD) systems that will support high performance data mining applications on geographically distributed data sources. In [6] is described an architecture, named *KNOWLEDGE GRID*, for PDKD systems that is built on top of computational grid services that provide dependable, consistent, and pervasive access to high-end computational resources. The *KNOWLEDGE GRID* architecture uses the grid services and defines a set of additional layers to implement the services of distributed knowledge discovery process on grid-connected sequential or parallel computers.

6 Conclusion

Applications in the area of data management and analysis show the highest growth rate among the applications developed on parallel computing machines. However, industrial, commercial, and government data mining success stories tend not to be publicized. Parallel and distributed data mining will play a more and more important role for data analysis and knowledge extraction in several application contexts analysis of scientific data mining of commercial, business and financial databases data extraction and decision support for government and public departments.

Data mining algorithms and underlying techniques can be parallelized to make them effective in the analysis of very large data sets. Several parallel strategies, algorithms, techniques, prototypes have been developed in the recent years. They allow researchers and end-users to mine large databases offering scalable performance. Nevertheless many promising research issues need to be faced and

interesting directions must be explored. Knowledge discovery is an area in which parallel computing can be used in a very profitable way. In fact, in this setting parallelism is exploited not only for quantitative computing, as occurs in many scientific computing applications, but also for *qualitative computing*.

References

1. R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, *Proc. of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, 1994.
2. R. Agrawal and J.C. Shafer, Parallel Mining of Association Rules, *IEEE Transactions on Knowledge and Data Engineering*, 8, 1996.
3. M.J.A. Berry and G. Linoff, *Data Mining Techniques for Marketing, Sales, and Customer Support*, Wiley Computer Publishing, 1997.
4. J.P. Bigus, *Data Mining with Neural Networks*, McGraw-Hill, New York, 1996.
5. M. Bruynooghe, Parallel Implementation of Fast Clustering Algorithms, *Proc. Int. Symp. On High Performance Computing*, pp. 65-78, 1989.
6. M. Cannataro, D. Talia and P. Trunfio, KNOWLEDGE GRID: High Performance Knowledge Discovery Services on the Grid, *Proc. 2nd Int. Workshop GRID 2001*, Denver, CO, LNCS 2242, Springer-Verlag, pp. 38-50, November 2001.
7. D. Foti, D. Lipari, C. Pizzuti and D. Talia, Scalable Parallel Clustering for Data Mining on Multicomputers, *Proc. of the 3rd Int. Workshop on High Performance Data Mining HPDM00-IPDPS*, Cancun, LNCS 1800, pp. 390-398, Springer-Verlag, 2000.
8. A.A. Freitas and S.H. Lavington, *Mining Very Large Database with Parallel Processing*, Kluwer Academic Publishers, 1998.
9. E.-H. Han, G. Karypis and V. Kumar, Scalable Parallel Data Mining for Association Rules, *IEEE Transactions on Knowledge and Data Engineering*, 1999.
10. D. Judd, K. McKinley and A.K. Jain, Large-Scale Parallel Data Clustering, *Proc. Int. Conf. On Pattern Recognition*, Vienna, 1996.
11. R. Kufrin, Generating C4.5 Production Rules in Parallel, *Proc. 14th Nat. Conf. on Artificial Intelligence - AAAI-97*, AAAI Press, 1997.
12. X. Li and Z. Fang, Parallel Clustering Algorithms, *Parallel Computing*, 11, pp. 275-290, 1989.
13. F. Neri and A. Giordana, A Parallel Genetic Algorithm for Concept Learning, *Proc. 6th Int. Conf. Genetic Algorithms*, pp. 436-443, 1995.
14. C.F. Olson, Parallel Algorithms for Hierarchical Clustering, *Parallel Computing*, 21, pp. 1313-1325, 1995.
15. R.A. Pearson, A Coarse-grained Parallel Induction Heuristic, in: H. Kitano, V. Kumar, C.B. Suttner (Eds.), *Parallel Processing for Artificial Intelligence 2*, Elsevier Science, pp. 207-226, 1994.
16. J.Shafer, R. Agrawal and M. Mehta, SPRINT: A Scalable Parallel Classifier for Data Mining, *Proc. 22nd Int. Conf. Very Large Databases - VLDB-96*, Bombay, 1996.
17. D. Skillicorn, Strategies for Parallel Data Mining, *IEEE Concurrency*, 7:4, pp. 26-35, 1999.
18. M.J. Zaki, Parallel and Distributed Association Mining: A Survey, *IEEE Concurrency*, 7:4, pp. 14-25, 1999.