F.S.A.I., Vol. 10, Nos. 1–2, pp. 67– 79, 2004

The Distributed Algorithms in Mining Association Rules

Cornelia Győrödi, Robert Győrödi, Alina Bogan-Marta, Mirela Pater

Department of Computer Science, Faculty of Electrotehnics and Informatics
University of Oradea, Romania
E-mail: cgyorodi@uoradea.ro, rgyorodi@rdsor.ro,
alinab@uoradea.ro, mirelap@uoradea.ro

Abstract: With the ever-growing database sizes, we have enormous quantities of data, but unfortunately we cannot use raw data in our day-to-day reasoning/decisions. We desperately need knowledge. This knowledge is in most cases in the gathered data, but the extraction of it is a very time and resources consuming operation. Association rule mining is a central problem in discovering knowledge. It finds interesting association or correlation relationships among a large set of data items. The paper presents some considerations about distributed association rules mining together with a comparison between three representative distributed algorithms, namely CDA, FDM and DDM. The compared algorithms are presented together with some experimental data that leads to the final conclusions.

1 Introduction

The continuing growth of database sizes and mainly the stringent need of pertinent information for decision support increased the interest in automatic knowledge discovery in databases.

The implicit information within databases, and mainly the interesting association relationships among sets of objects, that lead to association rules, may disclose useful patterns for decision support, financial forecast, marketing policies, even medical diagnosis and many other applications. This fact has attracted a lot of attention in recent data mining research [5]. As shown by Agrawal and Shrikant [1], mining association rules may require iterative scanning of large databases, which is costly in processing. Many authors focussed their work on efficient mining of association rules in databases [1], [2], [4], [5], and [6].

Association rule mining finds interesting association or correlation relationships among a large set of data items [6]. The association rules are considered interesting if they satisfy both a *minimum support* threshold and a *minimum confidence* threshold [3].

A very influential association rule mining algorithm, Apriori ([1]), has been developed for rule mining in large transaction databases. Many other algorithms developed are derivative and/or extensions of this algorithm. A large step forward in improving the performances of these algorithms was made by introduction of a novel, compact data structure, called *frequent pattern tree*, or FP-tree [2], and the associated mining algorithms, FP-growth (Han J. *et al.* 2000). Although these studies are on sequential data mining techniques, algorithms for parallel mining of association rules have also been proposed [7],[8]. The development of distributed algorithms for efficient mining of association rules has importance, based on the

following reasoning. (1) Databases may store a huge amount of data. Mining association rules in such databases may require substantial processing power, and a distributed system is a possible solution. (2) Many large databases are naturally distributed at different sites. Based on these observations, the study of efficient distributed algorithms for mining association rules is very important. Further more, a distributed mining algorithm can also be used to mine association rules in a single large database by partitioning the databases among a set of sites and processing the task in a distributed manner. This study assumes that the database to be studied is a transaction databases. The databases consist of a huge number of transaction records, each with a transaction identifier and a set of data items, and it is "horizontally" partitioned and allocated to the sites in a distributed system which communicates by message passing.

This paper presents a comparative study of the most important distributed algorithms used in association rules mining CDA (Count Distributed Algorithm) [8], FDM (Fast Distributed Mining of association rules) [7] and DDM (Distributed Decision Miner) [9]. It also includes a performance study that shows the advantages and disadvantage of the distributed algorithms CDA, FDM and DDM.

2 Problem Definition

Association rule mining finds interesting association or correlation relationships among a large set of data items [8]. The association rules are considered interesting if they satisfy both a *minimum support* threshold and a *minimum confidence* threshold [9].

A more formal definition is the following [4]. Let $\Im$ = {i1, i2, …, im} be a set of items. Let D, the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq \Im$. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is implication of the form $A \Rightarrow B$, where $A \subset \Im$, $B \subset \Im$, and $A \cap B = \varnothing$. The rule $A \Rightarrow B$ holds in the transaction set D with *support s*, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., both A and B). This is taken to be the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has *confidence c* in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(B|A). That is,

$$\text{support}(A \Rightarrow B) = P(A \cup B) \tag{1}$$

$$\text{confidence}(A \Rightarrow B) = P(A \mid B) \tag{2}$$

The definition of a frequent pattern relies on the following considerations [5]. A set of items is referred to as an itemset (pattern). An itemset that contains k items is a k-itemset. The set {*name, semester*} is a 2-itemset. The occurrence frequency of

an itemset is the number of transactions that contain the itemset. This is also known, simply, as the frequency, support count, or count of itemset. An itemset satisfies *minimum support* if the occurrence frequency of the itemset is greater than or equal to the product of *minimum support* and the total number of transactions in D. The number of transactions required for the itemset to satisfy *minimum support* is therefore referred to as the minimum support count. If an itemset satisfies *minimum support*, then it is a *frequent itemset (frequent pattern)*.

The basic problem definition of association rules mining is given in [1]. Here are presented only the aspects that are specific to association rules mining in a distributed environment. Let DB a transactional database with D transactions. Suppose that there are n sites $S_1, S_2,..., S_n$ in a distributed system and that the database is correspondently partitioned on the n sites into $\{DB^1, DB^2,..., DB^n\}$. Let D_i denote the dimension of partition DB^i for $i = 1,..., n$. Also let $X.sup$ and $X.sup_i$ denote the support factor of the itemset X from DB and respectively DB^i. $X.sup$ is known as the global support counter, and $X.sup_i$ as the local support counter for site i. For a given minimum support s, X is globally frequent if $X.sup \geq s \times P$, correspondently, X is locally frequent at site S_i, if $X.sup_i \geq s \times P_i$. In the following, L will correspond to globally frequent itemsets from DB, and $L_{(k)}$ to globally frequent k-itemsets from L. The main goal of a distributed association rules mining algorithm is finding the globally frequent itemsets L.

3 Distributed Algorithms In Association Rules Mining

According to Dunham [4] most parallel or distributed association rule algorithms strive to parallelize either the data, known as *data parallelism*, or the candidates, referred to as *task parallelism*. With task parallelism, the candidates are partitioned and counted separately at each processor. Obviously, the partition algorithm would be easy to parallelize using the task parallelism approach. Other dimensions in differentiating the parallel association rule algorithms are the load-balancing approach used and the architecture. The data parallelism algorithms have reduced communication costs over the task, because only the initial candidates (the set of items) and the local counts must be distributed at each iteration. With task parallelism, not only the candidates but also the local set of transactions must be broadcast to all other sites. However, the data parallelism algorithms require that memory at each processor be large enough to store all candidates at each scan (otherwise the performance will degrade considerably because I/O is required for both the database and the candidate set). The task parallelism approaches can avoid this because only the subset of the candidates that are assigned to a processor during each scan must fit into memory. Since not all partitions of the candidates must be the same size, the task parallel algorithms can adapt to the amount of memory at each site. The only restriction is that the total size of all candidates be small enough to fit into the total size of memory in all processors combined. Performance studies have shown that the data parallelism tasks scale linearly with the number of processors and the database size. Because of the reduced memory

requirements, however, the task parallelism may work where data parallelism may not work.

The reduction of the database scans is one of the main problems association rules mining algorithms take in consideration. Another problem in the distributed environment is the communication complexity, which has a great influence mainly if the number of sites is high and the bandwidth is limited. According to Schuster and Wolff, the most important factors in the communication complexity of distributed association rules mining (DARM) algorithms are the number of partitions *n*, and |C|, the number of itemsets considered throughout the algorithm.

3.1. The CDA Algorithm

One data parallelism algorithm is the *Count Distribution Algorithm (CDA)*. The database is divided into n partitions, one for each processor. Each processor counts the candidates for its data and then broadcasts its counts to all other processors. Each processor then determines the global counts. These then are used to determine the large itemsets and to generate the candidates for the next scan. The algorithm according to Dunham [4] is shown below.

```
Input:
I // itemsets
p^1, p^2, …p^n  //processors
DB = DB^1, DB^2, …DB^n //database divided into partitions
s //support
Output:
L //large itemsets
Count distribution algorithm:
Perform in parallel at each processor p^j;
//count in parallel
k = 0;      //k is used as the scan number
L = ∅;
C_1 = I; //initial candidates are set to be the items
repeat
   k = k + 1;
   L_k = ∅;
   for each I_i ∈ C_k do
         c^1_i = 0; //initial counts for each itemset are 0
        for each t_j ∈ DB^j do
             for each I_i ∈ C_k do
                  if I_i ∈ t_i then
                    c^1_i = c^1_i +1;
   broadcast c^1_i to all other processors;
   for each I_i ∈ C_k do //determine global counts
        c_i = Σ^p_{l=1} = c^1_i;
        for each I_i ∈ C_k do
             if c_i ≥ (s × | DB^1 ∪ DB^2 ∪ … ∪ DB^n|) then
                   L_k = L_k ∪ I_i;
```

```
 L = L ∪ Lk;
Ck+1 = Apriori-Gen(Lk)
until Ck+1 = ∅.
```

3.2. The FDM Algorithm

The FDM (Fast Distributed Algorithm for Data Mining) algorithm, proposed in [7] has the following distinguishing characteristics:

1. Candidate set generation is Apriori-like. However, some interesting properties of locally and globally frequent itemsets are used to generate a reduced set of candidates at each iteration, this resulting in a reduction in the number of messages interchanged between sites.
2. After the candidate sets were generated, two types of reduction techniques are applied, namely a local reduction and a global reduction, to eliminate some candidate sets from each site.
3. To be able to determine if a candidate set is frequent, the algorithm needs only $O(n)$ messages for the exchange of support counts, where n is the number of sites from the distributed system. This number is much less than a direct adaptation of Apriori, which would need $O(n^2)$ messages for calculating the support counts.

The algorithm according to Cheung [7] is shown below.

```
Input:
DB^i     //database partition at each site S_i
Output:
L //set of all globally large itemsets
Algorithm:
Iteratively execute the following program fragment (for the
k^th iteration) distributively at each site S_i. The algorithm
terminates when either L(k) = ∅, or the set of candidate
sets CG(k) = ∅.
if k = 1 then
   T_i(1) = get_local_count(DB^i, ∅, 1)
else {
   CG(k) = ∪^n_i=1 CG_i(k) = ∪^n_i=1 Apriori_gen(GL_i(k-1))
   T_i(k) = get_local_count(DB^i, CG(k), i) }
for each X ∈ T_i(k) do
   if X.sup_i ≥ s x D^i then
        for j = 1 to n do
             if polling_site(X) = S_j then
                  insert ⟨X, X.sup_i⟩ into LL_i,j(k)
for j = 1 to n do
   send LL_i,j(k) to site S_j
for j = 1 to n do {
   receive LL_j,i(k)
   for each X ∈ LL_j,i(k) do {
        if X ∉ LPi(k) then
```

```
            insert X into LP_i(k)
        update X.large_sites } }
for each X ∈ LP_i(k) do
   send_polling_request(X);
reply_polling_request(T_i(k))
for each X ∈ LP_i(k) do {
   receive X.sup_j from sites S_j
   where S_j ∉ X.large_sites
        X.sup = Σ^n_i=1 X.sup_i
   if X.sup ≥ s × D then
        insert X into G_i(k) }
broadcast G_i(k)
receive G_j(k) from all other sites S_j, (j ≠ i)
L_(k) = ∪^n_i=1 G_i(k)
divide L_(k) into GL_i(k), (I = 1,…,n)
return L_(k).
```

3.3. The DDM Algorithm

The DDM (Distributed Decision Miner) algorithm, proposed in [9] is in fact a basic algorithm for a family of distributed association rules mining algorithms. The basic idea, according to Schuster and Wolff, is to verify that an itemset is frequent before collecting its support counts from all parties. The algorithm differs from FDM in that the fact that an itemset is locally frequent in one partition is not considered sufficient evidence to trigger the collection of all the support counts for that itemset. Instead, the parties perform some kind of negotiation by the end of which they are able to decide which candidate itemsets are globally frequent and which are not. In this way no communication is wasted on globally infrequent, but locally frequent itemsets.

The negotiation is based on some hypothesises. There is a common hypothesis *H*, shared by all sites, concerned with the global support of every candidate itemset, which based on the local support counts for an itemset will correctly predict whether it is frequent or not. Beside the common hypothesis, a private one *P* is also computed, based on the support counts already expressed and the site's local support count for the candidate itemset.

The messages the parties send to one another contain pairs $\langle i, x_i^j \rangle$, where i is an itemset number and $x_i^j = Support(X_i, DB^j)$. There is the assumption that *j*, the origin of the message, can be inferred with negligible communication costs. For each party *p* and itemset X_i, let $G^p(X_i)$ be the group of all x_i such that $\langle i, x_i^j \rangle$ was received by *p*. An other assumption is that $G^p(X_i)$ is equal for all *p* and refer to it as $G(X_i)$, $\underline{D}$ is either known to all parties in advance or can be exchanged in the first *n* messages.

The algorithm according to Schuster and Wolff [9] is shown below (1):

$$H(X_i)=\begin{cases} 0 & \text{if } G(X_i)=\varnothing \\ \dfrac{\sum_{x_i^p \in G(X_i)} x_i^p}{\sum_{x_i^p \in G(X_i)} D^p}\cdot D & otherwise \end{cases} \qquad (1)$$

$$P(X_i, DB^j)=\sum_{x_i^p \in G(X^i)} x_i^p + \frac{x_i^j}{D^j}\cdot \sum_{x_i^p \notin G(X_i)} D^p$$

For node j out of n:

1. Initialize $C_1 = \{\{i\} : i \in I\}$, $k = 1$, $Passed = \varnothing$
2. **while** $Ck \neq \varnothing$
   a) do
   - Choose an itemset $X_i \in C_k$ which was not yet chosen and for which either $H(X_i) < MinFreq \cdot D \leq P(X_i, DB^j)$ or $P(X_i, DB^j) < MinFreq \cdot D \leq H(X_i)$, and broadcast $\langle i, Support(X_i, DB^j)\rangle$.
   - if no such itemset exists then broadcast ⟨pass⟩.

   b) **until** $|Passed| = n$.
   c) $L_k = \{X_i \in C_k : H(X_i) \geq MinFreq \cdot D\}$.
   d) Broadcast the support counts for every $X_i \in L_k$ *that was never chosen.*
   e) $C_{k+1} = Apriori\_Gen(L_k)$.
   f) $k=k+1$.
3. *Gen_Rules* $(L_1, L_2, \ldots, L_k)$

When node *j* receives a message *M* from node *p*:

1. **if** M = ⟨pass⟩ **then** insert p into Passed
2. **else if** |Passed| = n **then** M is the support counts of itemsets p has not yet sent. Update accordingly.
3. **else** $M = \langle i, Support(X_i, DB^p)\rangle$
   - **if** p ∈ Passed **then** remove p from Passed
   - Recalculate $H(X_i)$ and $P(X_i, DB^j)$

4 Comparative Study

The data set used for testing the performance of the three algorithms, CDA, FDM and DDM, was generated according to [1], by setting the number of items N=100, and the maximum number of frequent itemsets |L| = 3000, also the mean dimension of a transaction |T| = 10. To test the described algorithms 2 to 5 workstations with the following configuration: Pentium 4 at 1.5GHz, 512 MRAM, Windows 2000 Professional OS and 100Mb Ethernet network were used. The algorithms were implemented in Java 1.4, using the comparison test-bed for association rules mining algorithms, presented in Győrödi R. 2003, which was

extended to a distributed environment. To study the algorithms the support factor was varied between 0.5% and 40%.

A first result, obtained by testing the three algorithms on data sets with 50000 to 520000 transactions and, as mentioned before, using between 2 and 5 workstations with a support factor of 5% is shown in Figure 1. This figure shows that the performance of the algorithm depends on the number of processors and the number of transactions. For a data set with 520000 transactions that was distributed on two workstations, the execution time for the CDA algorithm was 5583 seconds, for the FDM was 4055 seconds and for DDM was 3486 seconds, while the same data set distributed on five workstations produced an execution time of just 1161 seconds for the CDA algorithm, 892 seconds for the FDM algorithm and 766 seconds for DDM. So, in order to obtain fairly small execution times, we need to increase the number of processors if we increase the number of transactions of the data set.

For a relatively small data set distributed on a large number of workstations the execution time could be quite long because the local sets of candidates obtained for each site is large and the communication time between sites becomes considerable, as well. For example for a data set with 110000 transactions distributed on 2 sites the execution time for the CDA algorithm was 88 second, for the FDM algorithm 68 seconds and for the DDM algorithm was 72 seconds, while the same data set distributed on 5 sites the execution time has risen to 120 seconds for the CDA algorithm, to 88 seconds for the FDM algorithm and to 92 seconds for the DDM algorithm. Thus, when increasing the number of sites (processors) the dimension of the data set must be taken into account. For a relatively small data set an increase in number of processors could lead to large sets of local candidates and a large number of messages transmitted between sites, thus, leading to an increase in execution time for the CDA, FDM and DDM algorithms.

Figure 1 shows that the performance of the algorithms increases with the number of processors, but the DDM algorithm has better performance than the FDM and CDA algorithms for the same number of processors and the same dimension of the data set, and a support factor of 5%.

Figure 2 show that the DDM, CDA and FDM algorithms present a good scalability relative to the different support factors, for a large data set with 520000 transactions. The performance of the algorithms increases with the support factor. Also if the number of processors is increased, the performance is quite good for a small support factor of only 5%. So, for a data set with 520000 transactions distributed on 5 workstations and a support factor of 5% the execution time for the DDM algorithm is 766 seconds, for the FDM algorithm is 892 seconds and for the CDA algorithm is 1161 seconds, meanwhile, for the same data set distributed on 2 workstations, the execution times increased five times, reaching 3486 for the DDM algorithm, 4055 for the FDM algorithm and 5583 for the CDA algorithm.

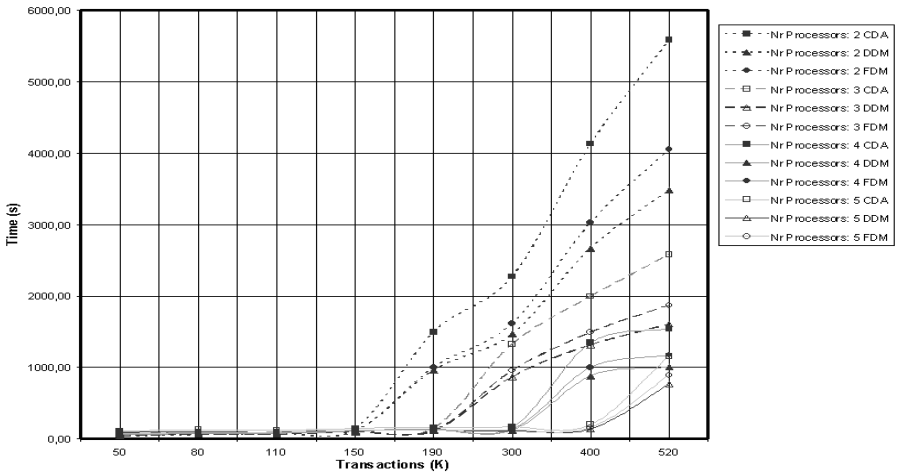

Fig.1 – Scalability by transactions and number of processors (sites) (5% suport)

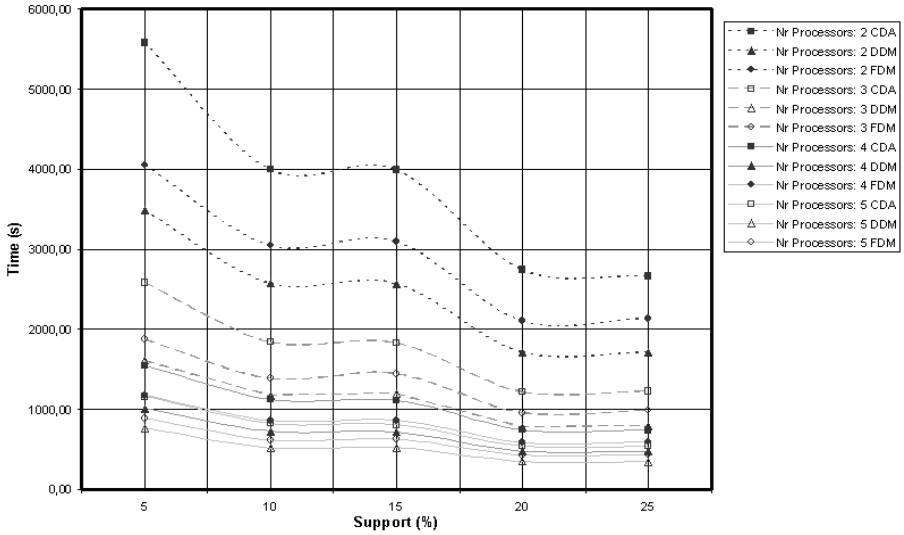

Fig.2 – Scalability by support and processors (sites) (D1 520K)

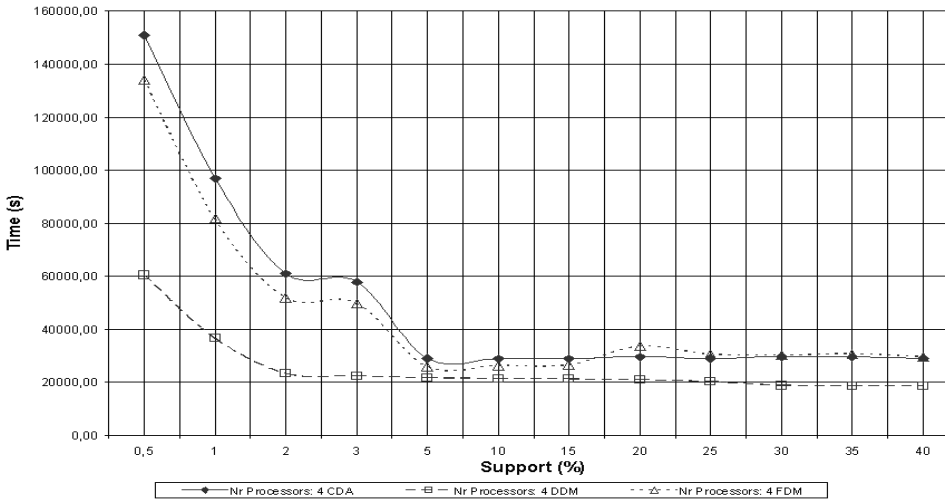

Fig.3 – Scalability by support (D1 10M)

Thus, in order to increase the performance and shorten execution times of the three algorithms for large data sets with small support factors is necessary to increase the number of processors.

The comparison of the performance of the CDA, FDM and DDM algorithms for the same data set distributed on 4 sites and for different support factors is shown in Figure 3. It is noticeable that the performance of the algorithms increases with the support factor, but the DDM algorithm presents a better performance than the FDM and CDA algorithms, especially for lower support factors.

5 Conclusions

From the experiments made, resulted a good scalability for the DDM, FDM and CDA algorithms, relative to different support factors for a large data set. It is also noticeable that increasing the support factor also increases the performance of the algorithms. Also good performances were obtained when the support factor was low and the data set large, but the number of processors increased. These results show the fact that the increase in processor number should be done relative to the dimension of the data set. Thus, for a relatively small data set, the large increase in processor number can lead to large sets of local candidates and a large number of messages, thus increasing the execution time of CDA and FDM algorithms, but due to its approach the DDM algorithm performed well even in these conditions. The CDA algorithm has a simple synchronization scheme, using only one set of messages for every step, while the FDM algorithm uses two synchronizations and the same scheme as CDA. For the test the FDM algorithm generates a smaller candidate set and also uses a smaller number of messages than the CDA algorithm, thus leading to a smaller execution time for the FDM algorithm. The communication optimization for the FDM algorithm is done using polling-sites. If

the data is evenly distributed between sites, the CDA and FDM algorithms produce the same results, but if the data is not evenly distributed between sites, the performance of the FDM algorithm increases. For the test the DDM algorithm generates the smallest candidate set and also uses the smallest number of messages, thus leading to the smallest execution time for the DDM algorithm. The communication optimization for the DDM algorithm is based on the fact that the algorithm first verifies if an itemset is frequent before collecting its support counts from all parties. The fact that an itemset is locally frequent in one partition is not considered sufficient evidence to trigger the collection of all the support counts for that itemset. Instead, the parties perform some kind of negotiation by the end of which they are able to decide which candidate itemsets are globally frequent and which are not.

The distributed mining algorithms can be used on distributed databases, as well as for mining large databases by partitioning them between sites and processing them in a distributed manner. The high flexibility, the scalability, the small cost/performance ratio and the connectivity of a distributed system make them an ideal platform for data mining.

References

[1] Agrawal R. and Srikant R., "Fast algorithms for mining association rules in large databases". *Proc. of 20th Int'l conf. on VLDB*: 487-499, 1994.

[2] Han J., Pei J., Yin Y., "Mining Frequent Patterns without Candidate Generation". *Proc. of ACM-SIGMOD*, 2000.

[3] Gyorodi C. and Gyorodi R., "Mining Association Rules in Large Databases". *Proc. of Oradea EMES'02*: 45-50, Oradea, Romania, 2002.

[4] Dunham M. H.,"Data Mining. Introductory and Advanced Topics". Prentice Hall, ISBN 0-13-088892-3, 2003.

[5] Fayyad U.M., et al**.** (1996), "From Data Mining to Knowledge Discovery: An Overview", *Advances in Knowledge Discovery and Data Mining*:1-34, AAAI Press/ MIT Press, ISBN 0-262-56097-6, 1996.

[6] Han J. and Kamber M., "Data Mining Concepts and Techniques", Morgan Kaufmann Publishers, San Francisco, USA, ISBN 1558604898, 2001.

[7] Cheung D. W., Han J., Vincent T. Ng., and Ada W. Fu. (1996), "A fast distributed algorithm for mining association rules". In Proceedings of IEEE 4th International Conference on Parallel and Distributed Information Systems, pages 31-42, December, 1996.

[8] Agrawal R. and Shafer J. C., "Parallel mining of association rules: Design, implementation and experience". In IBM Research Report, 1996

[9] Schuster A. and Wolff R., "Communication-efficient distributed mining of association rules". In Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, pages 473-484, Santa Barbara, California, May 2001.

[10] Gyorodi R**.,** "A comparative study of iterative algorithms in association rules mining". In Studies in Informatics and Control, Volume 12 Number 3, Romania, 2003, ISSN 1220-1766, pp. 205-215.

Authors:

Cornelia Győrödi graduated in Computer Engineering the "Politehnica" University of Timisoara (Romania) in 1994 and received her PhD in Computer Science in 2003 from "Politehnica" University of Timisoara as well. Her current research interests include artificial intelligence, neural networks, database management systems, knowledge discovery in databases and data mining techniques in different domains such as databases. She participated in different TEMPUS financed projects. She is author/co-author of 4 books, and more than 40 published papers, in the aforementioned fields, many of them abroad. Mrs. Győrödi is a lecturer at University of Oradea, Romania.

Robert S. Győrödi graduated in Computer Engineering the "Politehnica" University of Timisoara (Romania) in 1994. He is a PhD in Computer Science since 2001. His main research interests are in: combination of image processing, artificial intelligence, neural networks and database management systems in providing a safer operating experience within existing and future operating systems. He participated in different TEMPUS financed projects and had much collaboration with private companies, most notably with the European Drinks Group, coordinating their IT projects. He is author/co-author of 3 books, and of over 45 published papers, many of them abroad. He was visiting lecturer since 2001 until 2004 at University of Limerick, Ireland. Currently Mr. Győrödi is an Assistant Professor at University of Oradea, Romania.

Alina Bogan-Marta graduated in Computer Science the "Babes-Bolyai" University of Cluj-Napoca (Romania) in 1997. She is doing a PhD in Computer Science from 2001. Her main research interests are in: information retrieval techniques, speech processing, artificial intelligence, neural networks and bioinformatics. She participated in different projects financed by external research groups affiliated to European Universities and SOCRATES program. She is author/co-author of 2 didactic books, and over 15 published papers. Currently Ms. Bogan-Marta is lecturer at University of Oradea, and involved in research at Artificial Intelligence and Information Analysis Laboratory, at "Aristotle" University of Thessaloniki, Greece.

Mirela Pater graduated in Computer Engineering the "Traian Vuia" Polytechnics University of Timisoara (Romania) in 1985 and now is PhD student in Computer Science at "Politehnica" University of Timisoara. Her main research interests are in: database management systems, knowledge discovery in databases, data mining techniques in different domains such as databases and computer graphics. She participated in a SOCRATES program. She is author/co-author of 3 books, and over 20 published papers. Currently, Mrs. Pater is a lecturer at University of Oradea, Romania.