A Generalized Parallel Algorithm for Frequent Itemset Mining

MITICĂ CRAUS
"Gh. Asachi” Technical University
Department of Computer Engineering
53A D. Mangeron Street, 700050 Iasi
ROMANIA

ALEXANDRU ARCHIP
"Gh. Asachi” Technical University
Department of Computer Engineering
53A D. Mangeron Street, 700050 Iasi
ROMANIA

Abstract: A parallel algorithm for finding the frequent itemsets in a set of transactions is presented. The frequent individual items are identified by their index. We assume that processors number \( (m) \) is less than the frequent items number \( (n) \). At the first stage, every processor \( P_i, i \in \{1, \ldots, m-1\} \) sequentially computes the frequent itemsets from the interval \( I_i = [(i-1) \cdot p + 1, i \cdot p] \), where \( p = \lfloor \frac{n}{m} \rfloor \). The processor \( P_m \) computes frequent itemsets from the interval \( I_m = [(m-1) \cdot p + 1, n] \). In the second stage, the parallel algorithm is applied. The processor \( P_i \) computes, step by step, the sets \( F_{I_i} \) of the frequent itemsets with individual items from the intervals \( I_{i,j} = I_i \cup I_{i+1} \cup \ldots \cup I_j, j = i+1, \ldots, m \). In order to compute the set \( F_{I_i} \), the processor \( P_i \) uses \( F_{I_i} \) obtained in the previous step and \( F_{I_{i+1}} \) received from the processor \( P_{i+1} \). The main advantage of our parallel algorithm is that it uses a communication pattern known before algorithm start, which permits to map the communication to hardware. Another major advantage is that the set of the transactions can be distributed to processors before the beginning of the algorithm. This is possible because a processor \( P_i \) has to compute \( F_{I_i} \), \( j = i+1, \ldots, m \) and therefore only the transactions containing the frequent itemsets starting with \( I_i \) are needed.

Key–Words: Data Mining, Association Rule Discovery, Frequent Itemset Mining, Parallel Algorithms

1 Introduction

Association rule discovery (ARD) refers to the problem of finding all the relations between the elements of a data set so that the presence of a given element implies the presence of another element. An essential stage is the frequent itemset mining (FIM) in target databases. The target search space is exponential \( (2^n) \), where \( n \) is the number of the individual items in target database \( D \).

Let \( I \) be a set of individual items. A transaction is a set of distinct individual items (itemset) from \( I \). An itemset with \( k \) items is named \( k \)-itemset. The support of an itemset \( X \), denoted as \( \sigma(X) \), is the number or the percent \([1]\) of transactions which includes \( X \). An itemset is called frequent if its support is greater than a threshold \( \text{minimum support} \). A frequent \( k \)-itemset \( X \) is maximal if no other \( k' \)-itemset \( (k < k') \) contains \( X \) as a subset \([10]\).

The paper is organized as follows: In the second section there are presented the main results in the association rule mining field. The next two sections are dedicated to the base version of the new algorithm for the frequent itemset mining problem \([2]\) and its parallelization in the case when the processors number is equal to the frequent itemsets number \([3]\). The generalized algorithm is described in the fifth section. The last section contains some conclusions and unsolved or partially solved problems.

2 Related work

The Apriori algorithm is the first and one of the most important association rule mining algorithms. Apriori algorithm (Algorithm 1) was proposed by Agrawal and Srikant \([1]\).

At the first step, the Apriori algorithm determines all frequent 1-itemsets, by scanning the transaction database \( D \). At the subsequent \( k \)-th step, the candidate \( k \)-itemsets will first be generated, based on frequent \((k-1)\)-itemsets found in the corresponding \( k-1 \) step. Then, the support for each candidate \( k \)-itemset will be computed and the candidates that meet the minimum support condition will be retained.

An optimal generation of candidate \( k \)-itemsets is based on the Apriori Principle \([9]\): If an itemset is frequent, then all of its subsets must also be frequent. This theorem can easily be proven by analyzing the definition of support and considering the following property of the itemsets:

\[
\forall X, Y : X \subseteq Y \Rightarrow \sigma(X) \geq \sigma(Y)
\]

where \( X, Y \) are itemsets.
Also, Agrawal states that the results of Apriori are not influenced by imposing a particular order for the individual items prior to determining frequent itemsets [1].

**Algorithm 1 Apriori**

Notations:
- \( L_k \) = set of \( k \)-itemsets having minimum support (frequent itemsets)
- \( C_k \) = set of candidate \( k \)-itemsets (items to be counted)

Initial conditions:
- \( D \) = set of transactions
- \( L_1 \) = \{frequent 1-itemsets\}

Algorithm:

\[
\text{for } (k = 2; \text{ } L_{k-1} \neq \emptyset; k++) \text{ do} \\
\quad C_k \leftarrow \text{Apriori}_\text{gen} (L_{k-1}) \text{ /new candidates} \\
\qquad \text{for all } t \in D \text{ do} \\
\qquad \\
\qquad \quad C_t \leftarrow \text{Subset}(C_k, t) \text{ /candidates in } t \\
\qquad \quad \text{for all } c \in C_t \text{ do} \\
\qquad \\
\qquad \quad \quad c.\text{count}++ \\
\qquad \quad \text{end for} \\
\text{end for} \\
\text{end for} \\
L_k \leftarrow \{c \in C_k \mid c.\text{count} \geq \text{minsupp} \} \\
\text{end for} \\
\text{Answer} = \bigcup_k L_k
\]

Improved versions of the algorithm include Apriori-TID and Apriori-Hybrid [1], DHP [6], SON [7]. They improve the performances of the base algorithm through a series of modifications: reduce the number of database scans, reduce the size of the analyzed dataset at each scan, use of various pruning techniques.

The HPA (Hash Partitioned Apriori) algorithm is one of the most efficient parallel implementations of Apriori algorithm. This algorithm partitions both candidates and database transactions between processing nodes, using a hash function [8].

The speedup of the HPA algorithm is depending on the hash function. HPA avoids the memory overflow caused by a large number of candidate itemsets by partitioning those itemsets among nodes using hash function as in the hash join [8].

Apriori algorithm has two major drawbacks with respect to storage space, memory use and running time: multiple scans of the initial database and the complex phase of efficient candidate generation. In order to avoid the above mentioned issues, some alternative algorithms have been proposed.

FP-Growth (Frequent Pattern Growth) [5] is an algorithm which uses a tree structure [5] and determines all frequent itemsets with only two full scans of the target database, without candidate generation. The algorithm is composed of two phases: building the FP-Tree (Frequent Pattern Tree) and generation of frequent itemsets. Although it surpasses the performances of the Apriori algorithm, the FP-Growth algorithm is difficult to use within interactive systems within which users frequently modify support and confidence factors and within which database size is not constant.

Another tree based algorithm for mining association rules is RARM (Rapid Association Rule Mining) [4]. This algorithm employs tree structures in order to represent the input data sets.

### 3 Base algorithm for frequent itemset mining problem

Let us consider \( n \) individual frequent items. The fundamental idea of our algorithm for the FIM problem is to determine, step by step, the frequent itemsets, by enlarging the interval to which the individual items belong to. In the first step, the new candidates will be subsets of the sets \( \{1, 2\}, \{2, 3\}, \ldots, \{n - 1, n\} \). In the second step the new candidates will be subsets of the sets \( \{1, 2, 3\}, \{2, 3, 4\}, \ldots, \{n - 2, n - 1, n\} \). In the last step the new candidates will be subsets of the set \( \{1, 2, \ldots, n\} \). This approach simplifies the candidate generation function and has bigger parallelism potential than Apriori algorithm [2].

**Algorithm 2 Base algorithm for the FIM problem**

Initial conditions:
- \( D \) = set of transactions; \( F_{i,k} = \{\text{frequent item } i\} \)

Algorithm:

\[
\text{for } k = 1 \text{ to } n - 1 \text{ do} \\
\text{for all } i : i \in \{1, \ldots, n - k\} \text{ do} \\
\quad j \leftarrow i + k \\
\quad F_{i,j} \leftarrow F_{i,j-1} \cup F_{i+1,j} \\
\quad C_{i,j} \leftarrow \{X \cup Y \mid (X \in F_{i,j-1} \land i \in X) \land (Y \in F_{i+1,j} \land j \in Y)\} \\
\text{for all } t \in D \text{ do} \\
\quad C_t \leftarrow \text{subset}(C_{i,j}, t) \text{ /candidates in } t \\
\text{for all } c \in C_t \text{ do} \\
\quad \quad c.\text{count}++ \\
\text{end for} \\
\text{end for} \\
F_{i,j} = F_{i,j} \cup \{c \in C_{i,j} \mid c.\text{count} \geq \text{minsupp} \} \\
\text{end for} \\
\text{end for}
\]

The following notations are used:
\(F_{i,j} = \text{the set of all frequent itemsets from the interval } [i, j]\)
\(C_{i,j} = \text{the set of candidate itemsets from the interval } [i, j]\)

In [2] it is proved that the new frequent itemsets in \(F_{i,j}\) contain simultaneous the items \(i\) and \(j\). As consequence, the new candidates from the interval \([i, j]\) are obtained in accordance with the relation (2)

\[
C_{i,j} = \{X \cup Y \mid (X \in F_{i,j-1} \land i \in X) \land (Y \in F_{i+1,j} \land j \in Y)\} \tag{2}
\]

4 Parallelization of the base algorithm

The parallel algorithm is based on the fact that the frequent itemsets are built by sharing the work between \(m\) processors. We state that the processors number \((m)\) is equal to the frequent items number \((n)\). The sets \(F_{i,j}, j = i, \ldots, n\), are computed, in successive steps, by the processor \(P_i\). In order to compute the set \(F_{i,j}\), the processor \(P_i\) uses \(F_{i,j-1}\) from the previous step and \(F_{i+1,j}\) received from the processor \(P_{i+1}\).

The communication pattern is known before algorithm start. This is the main advantage. Another major advantage is that the set of the transactions can be distributed to processors prior to the beginning of the algorithm. This is possible because a processor \(P_i\) has to compute \(F_{i,j}, j = i, \ldots, n\) and therefore only the transactions containing the individual frequent item \(i\) are needed [3].

**Algorithm 3 Parallel algorithm for the FIM problem**

**Initial conditions:**
- \(D_i = \text{set of transactions assigned to } P_i\)
- \(F_{i,t} = \{\text{frequent item } i\}\)

**Algorithm:**

\[
\begin{align*}
\text{for } k = 1 \text{ to } n - 1 \text{ do} \\
\text{for all } i : i \in \{1, \ldots, n - k\} \text{ par do} \\
\quad j \leftarrow i + k \\
\quad F_{i,j} \leftarrow F_{i,j-1} \cup F_{i+1,j} \\
\quad C_{i,j} \leftarrow \{X \cup Y \mid (X \in F_{i,j-1} \land i \in X) \land (Y \in F_{i+1,j} \land j \in Y)\} \\
\text{for all transactions } t \in D_i \text{ do} \\
\quad C_t \leftarrow \text{subset}(C_{i,j}, t) \text{ /candidates in } t \\
\text{for all candidates } c \in C_t \text{ do} \\
\quad c.\text{count}++ \\
\text{end for} \\
\text{end for} \\
\text{end for} \\
\end{align*}
\]

5 Generalized parallel algorithm for frequent itemset mining problem

When the processors number \((m)\) is less than the frequent items number \((n)\), the parallel algorithm can be modified and generalized as follows:

In the first stage, every processor \(P_i, i \in \{1, \ldots, m - 1\}\) computes sequential the frequent itemsets from the interval \(I_i = [(i - 1) \cdot p + 1, i \cdot p]\), where \(p = \lfloor \frac{n}{m} \rfloor\). The processor \(P_m\) computes frequent itemsets from the interval \(I_m = [(m - 1) \cdot p + 1, n]\). Algorithm 2 can be used for this stage.

In the second stage, the generalized parallel algorithm for the FIM problem is applied.

**Algorithm 4 Generalized parallel algorithm for the FIM problem**

**Initial conditions:**
- \(D_i = \text{set of transactions assigned to } P_i\)
- \(F_{i,t} = \text{set of all frequent itemsets from } I_t\)

**Algorithm:**

\[
\begin{align*}
\text{for } k = 1 \text{ to } m - 1 \text{ do} \\
\text{for all } I_t : I_t \in \{1, \ldots, m - k\} \text{ do} \\
\quad j \leftarrow I_{t+k} \\
\quad F_{i,j} \leftarrow F_{i,j-1} \cup F_{i+1,j} \\
\quad C_{i,j} \leftarrow \{X \cup Y \mid (X \in F_{i,j-1} \land I_t \cap X \neq \emptyset) \land (Y \in F_{i+1,j} \land I_t \cap Y \neq \emptyset)\} \\
\text{for all transactions } t \in D_i \text{ do} \\
\quad C_t \leftarrow \text{subset}(C_{i,j}, t) \text{ /candidates in } t \\
\text{for all candidates } c \in C_t \text{ do} \\
\quad c.\text{count}++ \\
\text{end for} \\
\text{end for} \\
\text{end for} \\
\end{align*}
\]

The used notations are as follows:
- \(F_{i,j}\) = set of all frequent itemsets from the interval \(I_{i,j}\)
- \(C_{i,j}\) = set of candidate itemsets from the interval \(I_{i,j}\)

**Lemma 1** A frequent itemset from \(F_{i,j}\), which does not contain simultaneous subsets of items from \(I_i\) and \(I_j\), belongs to \(F_{i+1,j}\) or \(F_{i,j-1}\).

**Proof:** Let us suppose a frequent itemset \(C_i\) so that \(C_i \cap I_i = \emptyset\) is in \(F_{i,j}\), but not in \(F_{i+1,j}\). This means that \(F_{i+1,j}\) does not contains all frequent itemsets from the interval \(I_{i+1,j}\). Similarly, for an itemset \(C_j\) for which \(C_j \cap I_j = \emptyset\), let us suppose that it is in \(F_{i,j}\).
but not in $F_{i,j-1}$. This means that $F_{i,j-1}$ does not contain all frequent itemsets from the interval $I_{i,j-1}$.

\[ \square \]

**Lemma 2** The new frequent itemsets in $F_{i,j}$ contain simultaneous subsets of items from $I_i$ and $I_j$.

**Proof:** Lemma 2 is an immediate consequence of Lemma 1.

From Lemma 2, it results that the new candidates from the interval $I_{i,j}$ are obtained in accordance with the relation (3)

\[ C_{i,j} = \{ X \cup Y | (X \in F_{i,j-1} \land I_j \cap X \neq \emptyset) \land (Y \in F_{i+1,j} \land I_j \cap Y \neq \emptyset) \} \]  

(3)

---

**Figure 1:** Processors communication at the generalized parallel algorithm for the FIM problem

---

6 Conclusion and Future Work

Our algorithm determines, step by step, the frequent itemsets, by enlarging the interval to which the individual items belong to, unlike the Apriori algorithm, which increases by 1 the dimension of the new frequent itemsets with each passing step.

The parallel algorithm uses a communication pattern known before algorithm start. This is a very important issue because it permits hardware implementation of the processors’ communication pattern. Another major advantage is that the set of transactions can be distributed to processors before the beginning of the algorithm. Our algorithm is dynamically configurable, depending of the processor number and frequent individual items number.

The processors’ workload allocation still can be improved. This is one of the future objectives. The distribution of the transactions between the processors will be also in focus.

**Acknowledgements:** The Excellence Research Program, through grant 74 CEEX-II03 – ”Academic Grid for Complex Applications (GRAI)” has supported the research for this paper.

**References:**


