Efficient Mining of Association Rules in Oscillatory-based Data

Mohammad Saniee Abadeh *saniee@modares.ac.ir*
Assistant professor, Faculty of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, 14115-143, Iran

Mojtaba Ala *mojtaba.ala@gmail.com*
Computer Engineering Department
Payame Noor University
Tehran, Iran

Abstract

Association rules are one of the most researched areas of data mining. Finding frequent patterns is an important step in association rules mining which is very time consuming and costly. In this paper, an effective method for mining association rules in the data with the oscillatory value (up, down) is presented, such as the stock price variation in stock exchange, which, just a few numbers of the counts of itemsets are searched from the database, and the counts of the rest of itemsets are computed using the relationships that exist between these types of data. Also, the strategy of pruning is used to decrease the searching space and increase the rate of the mining process. Thus, there is no need to investigate the entire frequent patterns from the database. This takes less time to find frequent patterns. By executing the MR-Miner (an acronym for "Math Rules-Miner") algorithm, its performance on the real stock data is analyzed and shown. Our experiments show that the MR-Miner algorithm can find association rules very efficiently in the data based on Oscillatory value type.

Keywords: Data Mining, Association Rules, Frequent Patterns, Stock.

1. INTRODUCTION

Association rules mining is an important problem in the data mining filed which deals with exploring the association and hidden relationships between itemsets within a transaction [2]. For example, an association rule may be like this: "If the stock prices of Transfo and Parsian go down, at 80% of probability, the price of Iran Khodro goes down". This rule is seen in 40% of the transactions. In this example, the probability posed is called "confidence" and the percentage of the transactions cover this rule is called "support".

Several algorithms have been proposed for mining association rules. Among these, the best approaches are including: Apriori approach [1], [2], which was first introduced by Agrawal in 1993. This approach operates on the basis of the candidate generation; and, searching {there} is done through the method of breadth-first so that the network of itemsets is searched from one level to another one. Eclat's approach was presented by Zaki et al. in 1997 [5]. FP-growth approach was presented by Han et al. four years later [4]. In this approach (FP-growth), discovering frequent patterns is done without candidate generation through using FP-tree where searching is done by depth-first. Recently, several algorithms have also been introduced for this purpose such as [3], [12] which were presented by Borgelt et al.

Mining association rules is very useful in the financial issues and the capital market such as the stock market [7], [8], [9], [10], [11]. Data such as stock data include the oscillatory value type. In this paper, an effective method about the data with the oscillatory value type is presented in which, unlike the available algorithms that all frequent itemsets are mined from the database, just a few numbers of the counts of itemsets are searched from the database, and the counts of the

rest of itemsets are computed using the relationships that exist between these types of data. The method consists of two important stages. In the first stage, the frequencies of some patterns is searched via a database; and, in the second stage, the other frequent patterns are computed by the frequencies of the patterns which was obtained in the first stage.

The remainder of the paper is organized as follows: In section 2, issues related to mining association rules are described in the data with the oscillatory value type. In Section 3, the MR-Miner algorithm is introduced; the experiments results are presented in Section 4; and, Section 5 includes the conclusion of this paper.

2. DEFINITIONS

In this section, some concepts associated with mining association rules in the data with the oscillatory value type are described. Before doing so, we will first give some definitions.

Definition 1 Let $u(i)$ and $v(j)$ be two "items", in this case $u(i) = v(j)$, where $(i = j$ and $u = v)$.

Definition 2 Let $\alpha = \{u_0(i), u_1(i), \ldots, u_k(i)\}$ and $\beta = \{v_0(j), v_1(j), \ldots, v_k(j)\}$ be two "patterns" for $k \geq 1$. $\alpha = \beta$, where $u_m(i) = v_m(j)$ for $0 \leq m \leq k$.

Definition 3 In a transaction database D, the number of transactions is shown by $|D|$. Let α be a pattern, the frequency of α is shown by $count(\alpha)$ and the support of α is shown by $sup(\alpha) = count(\alpha)/|D|$. If $sup(\alpha)$ is less than the user-specified minimum support, "minsup", α will be called a "frequent pattern".

Definition 4 The number of items in a pattern is called the *length* of the pattern. A pattern with the length of k is called "k-pattern" (or k-itemset).
Example The length of $\{a(1), b(2), c(2), d(2)\}$ is equal to 4.

Definition 5 The number of items in a transaction is called the length of the transactions. The length of transactions is shown by L.

Definition 6 An association rule is like $\alpha \rightarrow \beta$, where: (1). Both α and $\alpha \cup \beta$ are frequent patterns; (2). $\alpha \cap \beta = \emptyset$; and, (3). $conf(\alpha \rightarrow \beta)$ is not less than the user-specified minimum confidence. The confidence of a rule is defined as $sup(\alpha \cup \beta) / sup(\alpha)$.

Oscillatory value: the value of an attribute in a transaction represents an increase or decrease in it. In the analysis of association on this type of data, only increase or decrease, are important in the numerical values showing the oscillation. Therefore, the numerical values turn into the "Up" or "Down".

TID/date	Attribute/stock symbol		
	ALBZ	**DLGM**	**INFO**
20100903	Up	Down	Up
20100904	Up	Down	Down
20100905	Down	Up	Down
20100906	Up	Up	Up

TABLE 1: Stock market data

Taking stock market database as an example, association rule mining can be used to analyze the share price movement**.** The stock prices variation in the stock market database is the oscillatory value type. As it is shown in Table 1, several trades are considered in the stock market database

of a transaction every day and the date of that day is used as its transaction ID (TID). Each attribute is the changes in stock price of a company accepted in a stock market which has two values: "Up" or "Down". "Up" indicates the increase of price in one stock, and "Down" indicates the decrease of price in one stock, which are shown briefly with the numbers of 1 and 2, respectively.

3. THE MR-MINER ALGORITHM

In this section, the MR-Miner algorithm is presented and the details of the method are shown step by step. The MR-Miner algorithm is made of the combination of four major functions of the Candidate, SeekDB, Prune2 and Compute. The Algorithm and its functions are shown in Figures 1, 3, 4, 6.

3.1. The MR-Miner Algorithm

To discover frequent itemsets, the main idea in this method is in such a way that: first, $(2^k - 1) * 2^{k-2}$ of the candidate itemsets are generated; then, the other 3 itemsets are generated by each of these k-itemsets for $k \geq 2$. There is a relationship between the counts of all obtained 4 itemsets so that by finding the frequency of 1 itemset from the database, the counts of the other 3 itemsets can be computed by Equation 6, with no need to the database. Of course, some of these candidate itemsets are pruned by the Prune2 function according to Figure 4.

```
Algorithm: MR-Miner(transaction database D, minsup)
        F = Candidate(D);
        F1 ← SeekDB(F1, D);
        F1 ← Compute(F1, |D|);
        For( k = 2; Fk≠ ∅; k++ )
            if(k>=3)
                Purn2(Fk, Fk-1);
            end if
            Fk=SeekDB(Fk, D);
            Fk ← Compute(Fk, Fk-1);
            for (each itemset i in Fk-1 )
               if (i.Count < minsup)
                   Delete i from Fk-1;
                end if
            end for
        F ← Fk;
        end for
Output All Frequent Patterns F;
```

FIGURE 1: The MR-Miner algorithm

In general, the MR-Miner algorithm consists of the following four phases.

PHASE I: Determination of candidate itemsets
PHASE II: Pruning based on the candidate frequent itemsets
PHASE III: Counting some itemsets via the database
PHASE IV: Computing the rest of itemsets, with no need to the database

The following presents the detailed method phase by phase.

3.2. PHASE I: Determination of Candidate Itemsets

First, the candidate patterns are generated in two stages: In the first stage, all itemsets $\alpha = \{u_0(i), u_1(i), \ldots, u_k(i)\}$ are generated for $i = 1$. To do this, 1-itemsets are generated by the available attributes in the transaction database corresponding to Figure 1; then, 2-itemsets are generated by joining 1-itemsets; similarly, the k-itemsets are generated by joining (k-1)-itemsets. All the generated itemsets include items with value of 1 (up). In the second stage, using each generated k-itemsets $(k \geq 2)$, $2^{k-2} - 1$ of the other itemsets are generated. To generate these

itemsets, the value of 2 is used in k-2 of their first items. The generated itemsets, in these two stages, are called candidate itemsets.

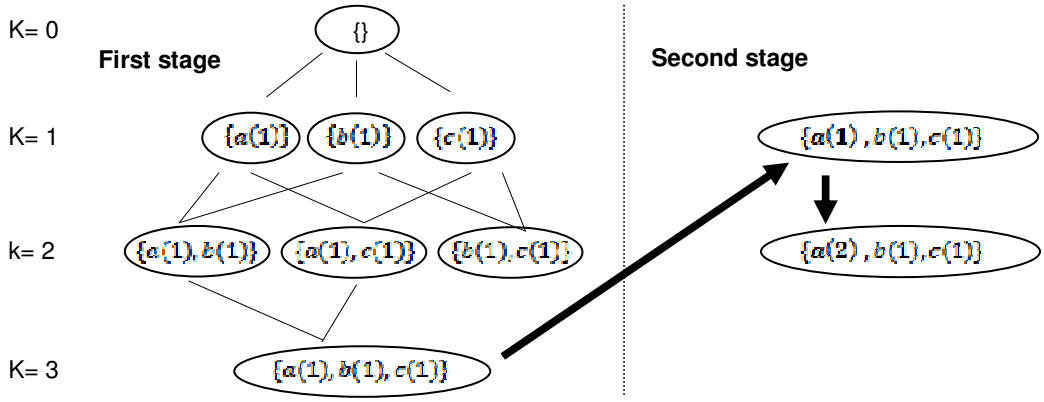

FIGURE 2: Generation of the candidate itemsets.

To generate candidate 1-itemsets $\{u(1)\}$, we need to have the attributes *u* from the database which are considered as the input of the Candidate function, and the Join function which is similar to the function of candidate generation in Apriori approach, has generated k-itemsets for $k \geq 2$ by joining (k-1)-itemsets. Finally, for each generated itemset, $2^{k-2}-1$ other itemsets are generated by $k-2$ of the first items. Figure 3 shows the Candidate function.

```
Function: Candidate(transaction database D)
        C0= ∅;
        for (each attribute a of D )
           Generate 1-Itemset c by a(1);
           C1 ←{c};
        end for
        for (k = 2; Fk-1 ≠ ∅; k++)
              Ck = Join(Ck-1);
        end for
        if (k>2)
          for (each Itemset c in Ck)
           Generate all c' by c: c ∋ c'={ui(j) | ∀ 1≤ i ≤ k-2
           then j is optional element from {1,2} and
           ∀ k-1≤ i ≤k then j=1}
             Ck ←{c'};
          end for
        end if
return Canadidate Patterns C;
```

FIGURE 3: The candidate itemset generation function

3.3. PHASE II: Pruning Based on the Candidate Frequent Itemsets

In the MR-Miner algorithm, the strategy of pruning is used to decrease the searching space and increase the rate of the mining process. Considering that all subsets of frequent k-itemsets are frequent, if the (k-2)-subset from any candidate frequent k-itemsets is not frequent, this candidate itemset and 2^2-1 other itemsets which should be made by that, will also not be frequent and will be deleted in this stage.

The code that is shown in Figure 4, with a survey of itemsets in the level of k-1 and k-2 delete a number of infrequent k-patterns and in this way the searching space will decrease.

```
Function: Prune2(k-Itemsets F_k, (k-1)-Itemsets F_{k-1}, minsup)
      for (each itemset i in F_k)
         find (k-2)-subset s of i in F_{k-2}
         if (s ∉ F_{k-2} or s.Count <= minsup)
            Delete i in F_k
          else
            find (k-1)-subset s of i in F_{k-1}
               if (s ∉ F_{k-1} or s.Count <= minsup)
                 Delete i in F_k
               end if
          end if
      end for
Return Patterns F_k
```

FIGURE 4: The Prune2 function

3.4. PHASE III: Counting Some Itemsets Via the Database

In this phase, some itemsets are counted via the database and their frequent patterns available are determined. This section of the algorithm, the database T and the candidate itemsets F_k with the length of *k* are received as input and the counts of patterns return as output. This is done by the SeekDB function. Similar to Apriori algorithm, this stage has searched the database once at any level.

3.5. PHASE IV: Computing the Rest of Itemsets, With no Need to the Database

The itemset $\{u(2)\}$ is generated for each candidate itemset $\{u(1)\}$ with the length of 1. To simplify, it can be assumed $x_1 = \{u(1)\}$, $x_2 = \{u(2)\}$.

Considering that each attribute *u* is one of the two values of increase or decrease, and the number of whole values of one attribute is equal to the number of the whole transactions, we have for each attribute *u*:

$$count\,(x_1) + count\,(x_2) = |D| \tag{1}$$

So, we have for each 1-itemset x_2:

$$count\,(x_2) = |D| - count\,(x_1) \tag{2}$$

So, to calculate the counts of the itemsets with the length of 1, all x_1 are searched from the database and all x_2 can be computed by Equation 2.
Also, for each candidate itemset with the length of higher than 1, there is $2^2 - 1$ other itemsets which should be generated, and their frequency should be computed.

For $k = 2$, $2^2 - 1$ other itemsets are generated by each of candidate 2-itemsets according to the Figure 5. There is a shared 1-itemset for all two 2-itemsets shown in Figure 5.

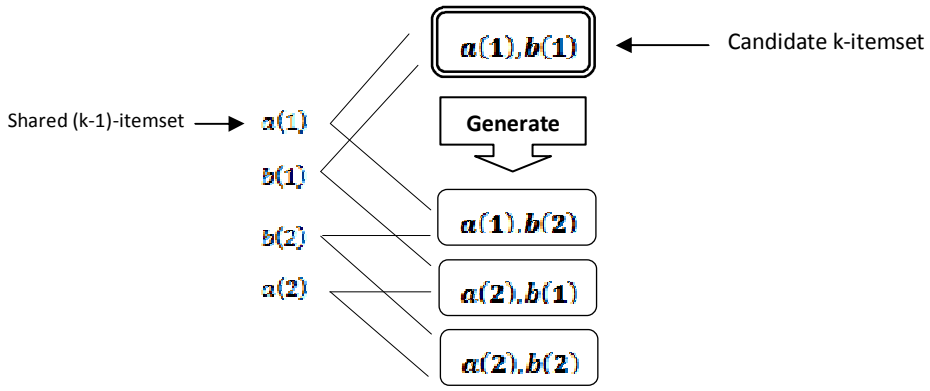

FIGURE 5: Relationship between 2-itemsets

Considering the available relationships in Figure 5, we have:

$$\begin{aligned} \{a(1),b(1)\} \cap \{a(1),b(2)\} &= \{a(1)\} \\ \{a(1),b(1)\} \cap \{a(2),b(1)\} &= \{b(1)\} \\ \{a(1),b(2)\} \cap \{a(2),b(2)\} &= \{b(2)\} \end{aligned} \tag{3}$$

We can extend Equation 3 for itemsets with the length more than 2. Let x_i be the itemset generated by a candidate itemset $x_1$with the length of k such that $i = 2,3,4$ and let y_j be the itemset with the length of $k-1$ such that $j = 1,2,4$. Regarding Equation 3, we have:

$$\begin{aligned} x_1 \cap x_2 &= y_1 \\ x_1 \cap x_3 &= y_2 \\ x_2 \cap x_4 &= y_4 \end{aligned} \tag{4}$$

So, for their counts, we have:

$$\begin{aligned} count(x_1) + count(x_2) &= count(y_1) \\ count(x_1) + count(x_3) &= count(y_2) \\ count(x_2) + count(x_4) &= count(y_4) \end{aligned} \tag{5}$$

Finally, to compute the counts of k-itemsets x_2, x_3 , x_4, we have:

$$\begin{aligned} count(x_2) &= count(y_1) - count(x_1) \\ count(x_3) &= count(y_2) - count(x_1) \\ count(x_4) &= count(y_4) - count(x_2) \end{aligned} \tag{6}$$

Thus, by any candidate k-itemsets $x_1 (k \geq 2)$, 3 k-itemsets x_i $(i = 2,3,4)$ are generated which, at every level, by searching the frequency of the itemset x_1 from the database, the counts of the itemsets x_2, x_3, x_4via Equation 6, are computed. Figure 6 shows the Compute function.

```
Function: Compute( F_k, F_{k-1}, |D|)
      for (each k-itemset i in F_k)
         x_1=i;
         if (k = 1)
            x_2.Count = |D| - x_1.Count;
            F_k ← x_2;
         else
            Find y_1,y_2,y_3 in F_{k-1}
            Generate x_2, x_3, x_4 by x_1
            x_2.Count = y_1.Count − x_1.Count;
            x_3.Count = y_2.Count − x_1.Count;
            x_4.Count = y_3.Count − x_2.Count;
            F_k ← x_2,x_3,x_4;
         end if
      end for
Return F_k
```

FIGURE 6: The Compute function

4. PERFORMANCE STUDY

We evaluated the performance of the MR-Miner algorithm on two synthetic datasets and three real datasets with various parameters. The data of Tehran Stock Exchange were collected from Tehran Securities Exchange Technology Management Company (TSETMC). We compared proposed algorithm with the Apriori algorithm. Apriori algorithm is a very useful and well-known algorithm and has been used in many articles related to mining association in exchange data in the recent years [8], [9], [10]. All experiments were running on a computer with a Pentium 3.06GHz processing, 1GB of main memory, 500GB of hard disk capacity and the operating system of Microsoft Windows XP. Both algorithms have been implemented using C# in the environment of Microsoft Visual Studio 2008.

Parameters	Real-life datasets			Synthetic datasets	
	Dataset 1	**Dataset 2**	**Dataset 3**	**Dataset 4**	**Dataset 5**
Number of transaction	2000	2000	2000	5000	10000
Length of transaction	6	10	16	10	10

TABLE 2: Sets of the parameters for 5 datasets

We generated two synthetic datasets, dataset 4 and dataset 5, using the parameters shown in Table 2, and, the first three are real stock data.

4.1. Experiment on Synthetic Data

In this section, we compare the MR-Miner algorithm with the Apriori Algorithm by varying one parameter, which maintaining the other parameter at the default values shown in Table 2.

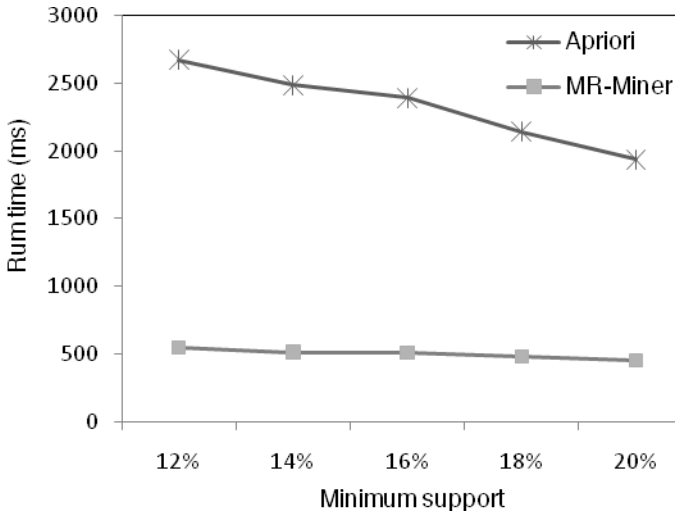

FIGURE 7: Minimum support versus run time, Dataset 4

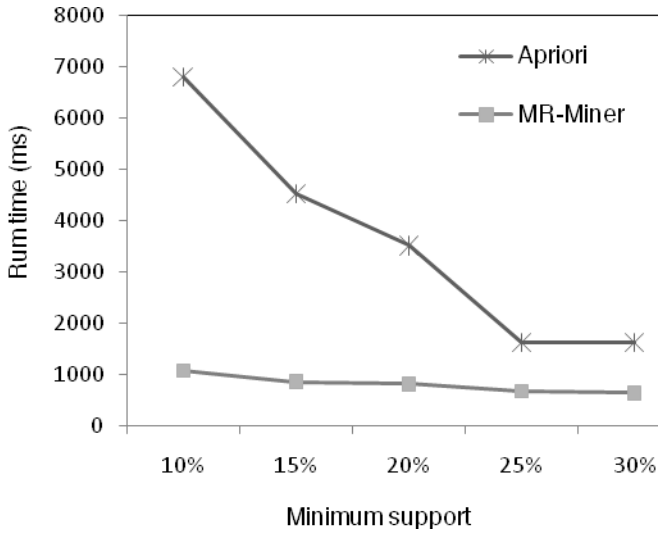

FIGURE 8: Minimum support versus run time, Dataset 5

Figures 7, 8 and 9 have shown the minimum support versus the run time. In Figure 8, the minimum support was investigated versus the run time with the lower minimum support for dataset 5. In this case, more itemsets were mined. By decreasing the amount of minimum support, the proposed algorithm had much less linear increase compared to the Apriori algorithm. Also, by increasing the minimum support, the run time on the dataset 4 is shown in Figures 7 and 9.

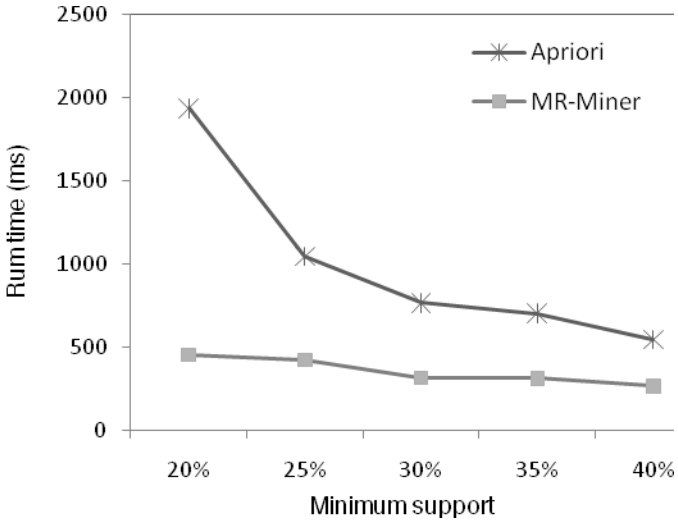

FIGURE 9: Minimum support versus run time, Dataset 4

Figure 10 shows the effect of increasing the number of transactions versus the run time on the dataset 2, 4 and 5. 2000, 5000 and 10000 were considered as the numbers of transactions. The minimum support of 20% was used for all datasets. The run time of both algorithms will increase versus increasing the number of transactions, so that the proposed algorithm will need 3 times less run time than the Apriori algorithm.

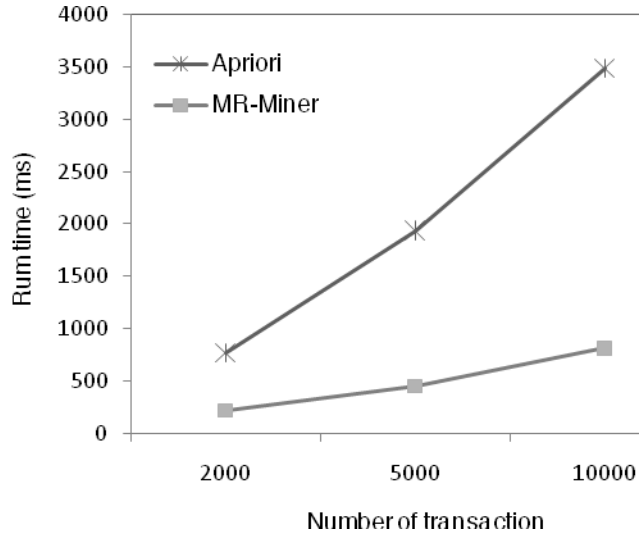

FIGURE 10: Number of transactions versus run time, with *minsup* = 20%

We also investigated the effect of the length of a transaction on both algorithms. The results are shown in Figure 11. In both algorithms, the run time had increased by increasing the length of transaction. Therefore, under the same support, more itemsets would be generated if the length of transactions increased. So, more candidate itemsets should be generated. In the proposed algorithm, since computing the entire frequent itemsets from database is not needed, by increasing the length of transactions, 3 times less time is needed than the Apriori algorithm.

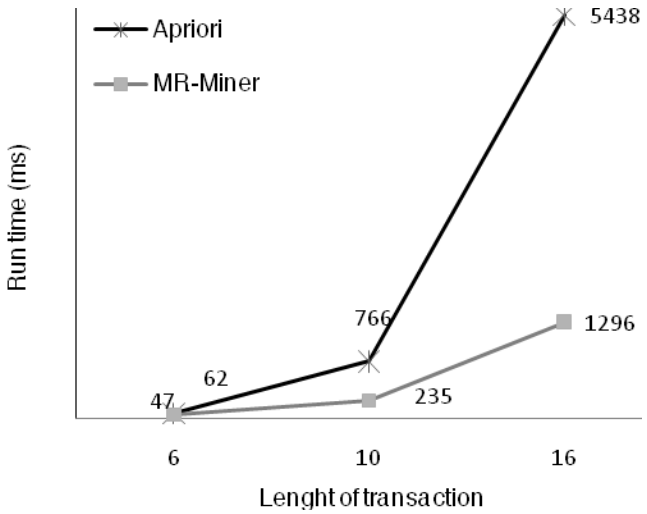

FIGURE 11: Length of transactions versus run time, with *minsup* = 16%

4.2. Experiments on Real Data

We now compare the MR-Miner algorithm and Apriori algorithm by mining three datasets 1, 2, 3. The datasets are included the stock price changes of eight companies: Iran Khodro, Jaber Ebne Hayyan Pharmacy, Tehran Cement, Mines & Metals Development, Isfahan Petrochemicals, Iran Transfo, Kashan Amirkabir Steel and Parsian Bank. The data were related to the trading days from March 2001 to June 2011.

Figures 12 to 14, illustrate the run time versus the minimum support for Dataset 2 and Dataset 3. The MR-Miner algorithm runs 3-4 times faster than the Apriori algorithm for real data.

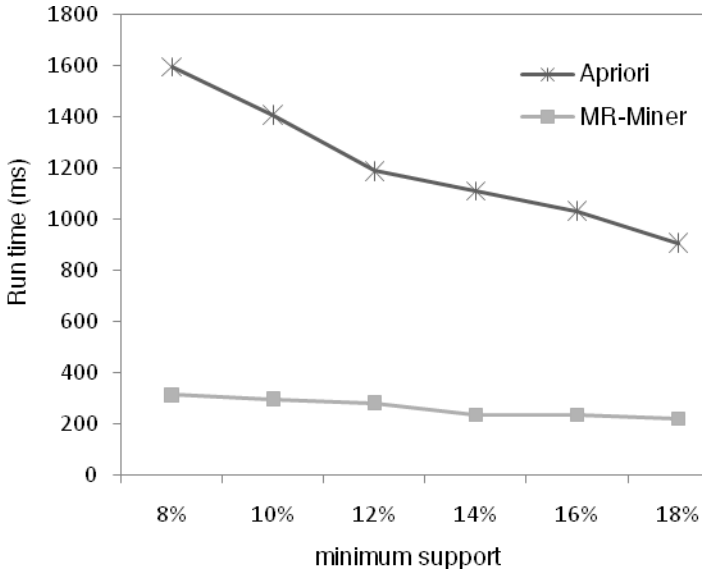

FIGURE 12: Minimum support versus run time, Dataset 2

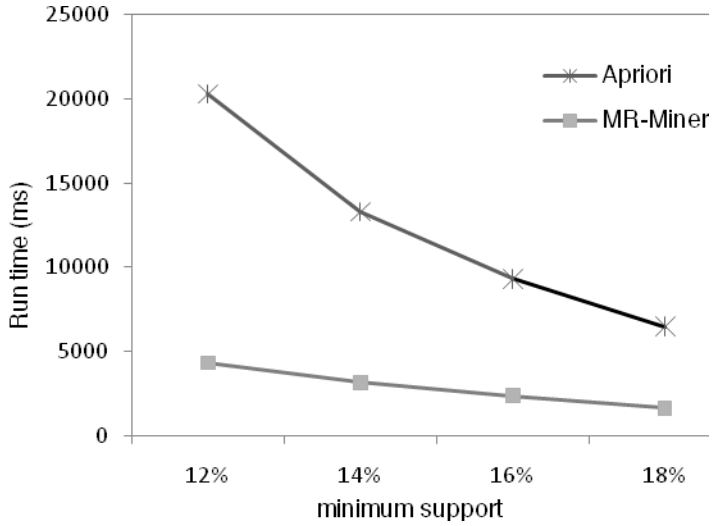

FIGURE 13: Minimum support versus run time, Dataset 3

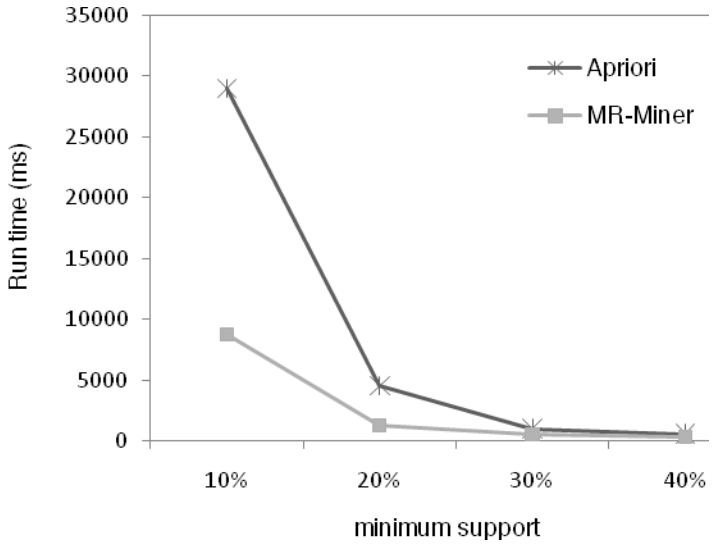

FIGURE 14: Minimum support versus run time, Dataset 3

In summary, the proposed algorithm was implemented faster than the Apriori algorithm on all datasets. The results had shown that the MR-Miner algorithm had operated better than the Apriori algorithm in all cases. Considering that the presented algorithm is an absolute algorithm and does not use the sampling methods, the accuracy of this algorithm likes that of the Apriori algorithm is 100%.

When mining the Tehran stock exchange data, some interesting rules were found. For Dataset 3, a sample rule found was “BTRNS(2), BPAR(2) → STEH(2)”. That is, if Transfo and Parsian fall, then Tehran Cement will fall with support =22% and confidence=75%. Also a sample rule found was “IKCO(1), BTRNS(2) → STEH(2)”. That is, if Iran Khodro rise, and Transfo falls, then Tehran Cement will fall with support=19% and confidence 67%. It should be noted that the association rules have a lower support on this type of attributes.

No	Rules	Support	Confidence
1	DBJR(1)--> STEH(2)	30%	61%
2	PESF(2)--> STEH(2)	33%	73%
3	BTRNS(2)--> STEH(2)	39%	66%
4	BPAR(2)--> STEH(2)	34%	71%
5	DBJR(1), PESF(1)--> IKCO(1)	21%	71%
6	BTRNS(2), FAJR(2)--> STEH(2)	24%	68%
7	BTRNS(2), BPAR(2)--> STEH(2)	22%	75%
8	FAJR(2), BPAR(2)--> STEH(2)	23%	75%
9	BTRNS(2), BPAR(2) --> MADN(2)	23%	77%
10	IKCO(1), DBJR(1)--> MADN(1)	19%	61%
11	DBJR(1), MADN(1)--> IKCO(1)	19%	75%
12	MADN(1), FAJR(1)--> IKCO(1)	18%	73%
13	PESF(1), MADN(1)--> BPAR(1)	18%	65%
14	MADN(1), FAJR(1)--> BPAR(1)	19%	75%
15	DBJR(1), STEH(2)--> IKCO(1)	18%	61%
16	IKCO(1), BTRNS(2)--> STEH(2)	19%	67%
17	PESF(1), STEH(2)--> BTRNS(2)	19%	63%

TABLE 3: Some of the association rules extracted from real data of Tehran stock exchange

Table 3 shows more association in decreasing the stock prices of the analyzed companies on the Tehran exchange data, such as rules 2, 3, 4, 6, 8 and 9 in Table 3 which can be seen with a more powerful support whereas the associations of stock price increase can be seen with a less support, such as rules 5, 10, 11, 12, 13 and 14.

5. CONCLUSION AND FUTURE WORK

In this paper, using the available relationship between the counts of itemsets inside the data with the oscillating value type, a new algorithm was implemented on the basis of generation of the candidate patterns. In the MR-Miner algorithm, based on relationship between the counts of itemsets in this type of data, many counts of itemsets were computed via the counts of other itemsets and with no need to searching the database. On the other hand, pruning strategy was used to reduce the searching space effectively. Thus, much less time was spent to find frequent patterns. By conducting experiments on the real data and synthetic data, it was shown that the presented algorithm had operated much faster than the basic Apriori algorithm.

Although we have shown that the proposed algorithm can efficiently mine association rules on the data with the oscillatory value type, there are still some issues that should be addressed in future researches, as we may extend the proposed algorithm from the frequent patterns to the closed frequent patterns. In order to have more decrease on the search space, a more effective function of prune can be implemented.

ACKNOWLEDGMENTS

We are grateful to Roholla Yosefian for his fruitful contributions on the mathematics issues and we are also grateful to our best friends Hamid Mohammadi and Fareed Mohammadi whose their help was encouraging.

REFERENCES

[1] R. Agrawal, T. Imielienski and A. Swami. “Mining Association Rules between Sets of Items in Large Databases”, in Proc. Management of Data, 1993, pp 207–216.

[2] R. Agrawal, R. Srikant, “Fast Algorithm for Mining Association Rules”, in Proc. VLDB, Santiago de chile, 1994.

[3] C. Borgelt and X. Yang, “Finding Closed Frequent Item Sets by Intersecting Transactions”, in Proc. 14th Int. Conf. on Extending Database Technology (EDBT 2011, Uppsala, Sweden), 2011.

[4] J. Han, H. Pei and Y. Yin, “Mining Frequent Patterns without Candidate Generation”. In Proc. the Management of Data (SIGMOD'00, Dallas, TX), New York, NY, USA, ACM Press, pp. 1-12, 2000.

[5] M. Zaki, S. Parthasarathy, M. Ogihara and W. Li, “New Algorithms for Fast Discovery of Association Rules”, In Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97, Newport Beach, CA), AAAI Press, pp. 283-296, 1997.

[6] Claudio Lucchese, Salvatore and Raffaele Perego, “Fast and Memory Efficient Mining of Frequent Closed Itemsets”, in Proc Knowledge and Data Engineering, pp. 21-36, 2006.

[7] Anthony J.T. Lee and Chun-Sheng Wang, “An Efficient Algorithm for Mining Frequent Inter-Transaction Patterns”, Information Sciences 177, pp. 3453-3476, 2007.

[8] Shu-hsien Liao, Pei-hui Chu and Tzu-kang Teng, “Mining the co-movement in the Taiwan Stock Funds Market”, Expert Systems with Applications, Volume 38, Issue 5, pp. 5276-5288, 2011.

[9] Sung Hoon Na and So Young Sohn, “Forecasting Changes in Korea Composite Stock Price Index (KOSPI) using Association Rules”, Expert Systems with Applications, Volume 38, Issue 7, pp. 9046-9049, 2011.

[10] Hulyi Tan Cai and H.J. Yong Li, “Frequent Patterns of Investment Behaviors in Shanghai Stock Market”, Science and Software Engineering, Volume 04, pp. 325-328, 2008.

[11] Vladimir Boginski, Sergiy Butenko and Panos M. Pardalos, “Mining Market Data: A Network Approach”, Computers & Operations Research, Volume 33, Issue 11, pp. 3171-3184, 2006.

[12] Borgelt, C., Yang, X., Nogales-Cadenas, R., Carmona-Saez, P., Pascual-Montano, A. “Finding Closed Frequent Item Sets by Intersecting Transactions”, In. 14th. Extending Database Technology, ACM Press, New York, pp. 367-376, 2011.