Mining Closed Multi-Dimensional Interval Patterns

Anthony J.T. Lee*
Department of Information Management, National Taiwan University

Fu-Chen Yang
Department of Information Management, National Taiwan University

Wei-Cheng Lee
Department of Information Management, National Taiwan University

Abstract

Many methods have been proposed to find frequent one-dimensional (1-D) interval patterns, where each event in the database is realized by a 1-D interval. However, the events in many applications are in nature realized by multi-dimensional intervals. Therefore, in this paper, we propose an efficient algorithm, called MIAMI, to mine closed multi-dimensional interval patterns from a database. The MIAMI algorithm employs a pattern tree to enumerate all closed patterns in a depth-first search manner. In the mining process, we devisethree effective pruning strategies to remove impossible candidates and perform a closure checking scheme to eliminate non-closed patterns. The experimental results show that the MIAMI algorithm is more efficient and scalable than the modified Apriori algorithm.

Keywords: multi-dimension interval pattern, 1-dimension interval pattern, frequent pattern, closed pattern, data mining

* Corresponding author. Email: jtlee@ntu.edu.tw
2011/03/19 received; 2011/07/07 revised; 2011/08/04 accepted

探勘封閉性多維度區間樣式

李瑞庭*
國立臺灣大學資訊管理學系

楊富丞
國立臺灣大學資訊管理學系

李偉誠
國立臺灣大學資訊管理學系

摘要

目前，已有許多學者提出探勘頻繁一維區間樣式的方法。但是，在實務上有許多應用包括多維度區間的資料。因此，在本篇論文中，我們提出「MIAMI」演算法，它利用頻繁樣式樹，以深度優先法遞迴產生所有的封閉性多維度區間樣式。在探勘的過程中，我們設計三個有效的修剪策略，以刪除不可能的候選樣式，以及使用封閉性測試移除非封閉性樣式。實驗結果顯示，MIAMI 演算法比改良式 Apriori 演算法更有效率，也更具擴充性。

關鍵詞：多維區間樣式、一維區間樣式、頻繁樣式、封閉性樣式、資料探勘

* 本文通訊作者。電子郵件信箱：jtlee@ntu.edu.tw
2011/03/19 投稿；2011/07/07 修訂；2011/08/04 接受

1. INTRODUCTION

Interval-based pattern mining can be applied to various domains such as meteorology, financial market analysis, bioinformatics, linguistics, and so on. Many methods (Amo et al. 2008; Guyet & Quiniou 2008; Höppner 2001; Kam & Fu 2000; Kong et al. 2010; Lee et al. 2009; Papapetrou et al. 2009; Patel et al. 2008; Winarko & Roddick 2007; Wu & Chen 2007; Wu & Chen 2009) have been proposed to find frequent one-dimensional (1-D) interval patterns, where each event in the database is realized by a 1-D interval. A pattern is frequent if its support is not less than a user-specified minimum support threshold, where the support of a pattern is defined as the number of transactions containing the pattern in the database.

However, the events in many applications are in nature realized by multi-dimensional (M-D) intervals. For example, a company may investigate its competences and weaknesses under many financial indices, such as liability ratio, operating profit ratio, earnings per share, marketing spending, and inventory cost, where each index during a certain period of time may be represented by a 1-D interval. If we find a pattern that the inventory cost of a company is greater than those of the other companies, this may indicate that the company needs a more effective delivery mechanism and inventory management. In addition, if we find another pattern that both the revenue and marketing spending of a company increase over time, this may suggest an expansion over marketing budget. However, these M-D interval patterns cannot be mined by the previously proposed algorithms. Those applications have motivated us to develop an algorithm of mining multi-dimensional interval patterns.

The frequent pattern mining problem was first introduced by Agrawaland Srikant (1994). They proposed an algorithm, called Apriori, which uses the antimonotone property to mine frequent itemsets from databases. The antimonotone property states that all subpatterns of a frequent pattern must be frequent. Based on this property, a frequent pattern of length k (k-pattern) can be generated by joining two frequent (k-1)-patterns, k>2. The Apriori algorithm performs well for sparse datasets but not well for dense datasets due to generating a large number of candidates and repeatedly scanning the database. Han et al. (2000) introduced the FP-Growth algorithm to avoid generating impossible candidates. The FP-Growth algorithm summarizes the database into a FP-tree and uses the pattern growth approach to mine frequent patterns. Since all information needed is stored in the FP-tree, it is not necessary to scan the database after the tree has been created. Zaki & Hsiao (2005) proposed an algorithm, called CHARM, to mine closed itemsets by using effective pruning strategies to remove impossible candidates during the mining process. A frequent pattern *P* is closed if there does not exist any super-patterns of *P* with the same support.

Sequential pattern mining discovers frequent subsequences as patterns in a sequence database, where each transaction contains a sequence of itemsets. After the problem was first introduced by Agrawal and Srikant (1995), many sequential pattern mining algorithms (Han et al. 2000; Pei et al. 2000; Zaki 2001) have been proposed. GSP (Agrawal & Srikant1995), an Apriori-based method, generates candidates and finds all frequent patterns level by level in a breath-first search manner. SPADE (Zaki 2001) uses a vertical id-list format to represent the database and utilizes it to discover all frequent patterns in the database. FreeSpan (Han et al. 2000) and PrefixSpan (Pei et al. 2001), based on the FP-Growth method (Han et al. 2000), build a projected database for each frequent pattern and use it to find frequent patterns in a depth-first search manner.

Unlike sequential pattern mining, interval-based pattern mining focuses on discovering relationships between events from the database where each transaction contains a sequence of 1-D intervals (events). Kam and Fu (2000) used Allen's representation (Allen 1983) to illustrate the relationship between two intervals and proposed a method, based on the Apriori algorithm, for finding frequent interval patterns level by level. However, there is an ambiguity problem on the pattern representation in Kam and Fu's method. To resolve this problem, Wu and Chen (2007) proposed a non-ambiguous pattern representation. Based on the representation, they modified the PrefixSpan algorithm, called TPrefixSpan, to mine frequent interval pattern in a depth-first search manner. Höppner (2001) proposed a method to discover temporal rules from temporal state sequence databases. Winarkoand Roddick (2007) introduced a maximum gap time constraint to get rid of insignificant patterns. Papapetrou et al. (2009) developed a method that mines temporal arrangements of event intervals. Lee et al. (2009) used an algorithm to summarize a database into a generalized database, which can be used to efficiently mine interval patterns. Guyet et al. (2008) utilized a density estimation of the distribution of events intervals to find frequent patterns from temporal sequences. Patel et al. (2008) proposed a method, called IEMiner, which finds frequent relational patterns and uses the patterns found to construct an interval-based classifier to classify sequences into the closely related classes. Amo et al. (2008) designed an Apriori-based algorithm, called MILPRIT, to mine interval-based and point-based patterns. Wu and Chen (2009) presented an algorithm, called HTPM, which finds temporal patterns from interval-based and point-based datasets. Kong et al.

(2010) proposed an algorithm with four temporal predicates and applied it to explore possible associative movements between the stock markets of Mainland Chinaand Hong Kong.

To the best of our knowledge, there is no algorithm specially designed to mine multi-dimensional interval patterns. Therefore, in this paper, we propose an algorithm, called MIAMI, to mine closed multi-dimensional interval patterns from a database. The MIAMI algorithm employs a pattern tree to enumerate all closed patterns in a depth-first search manner. During the mining process, we devise three effective pruning strategies to remove impossible candidates and perform a closure checking scheme to eliminate non-closed patterns. Thus, it can efficiently mine closed multi-dimensional interval patterns from the database.

The contributions of this paper are summarized as follows. First, we introduce a novel concept of multi-dimensional interval patterns. Second, we propose an efficient algorithm, called MIAMI, to mine closed multi-dimensional interval patterns from databases. Third, during the mining process, we employ three effective pruning strategies to prune impossible candidates and perform a closure checking scheme to remove non-closed patterns. Finally, the experimental results show that the proposed algorithm is more efficient and scalable than the modified Apriori algorithm.

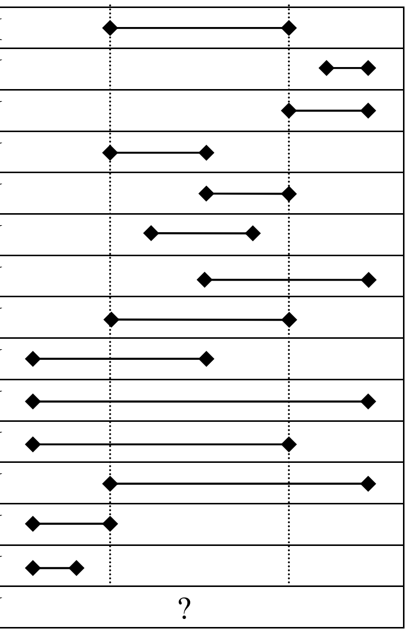
The rest of the paper is organized as follows. Section 2 introduces the preliminary concepts and problem definitions. Section 3 describes the proposed algorithm in detail and presents an example to demonstrate how it works. Section 4 shows the performance evaluation. Finally, the concluding remarks and future work are discussed in Section 5.

2. PRELIMINARIES AND PROBLEM DEFINITIONS

Consider a database containing *m* transactions, where each transaction contains a sequence of multi-dimensional intervals.

Definition 1. (Interval) $I = (P_s, P_e)$ is a one-dimensional (1-D) interval, where $P_s \leq P_e$. Note that if $P_s = P_e$, an interval will be degenerated into a point. That is, a point is a special case of an interval.

Definition 2. (Relationship) Given two 1-D intervals $I_1 = (P_{s1}, P_{e1})$ and $I_2 = (P_{s2}, P_{e2})$, the 1-D relationship between I_1 and I_2 is denoted as $IR(I_1, I_2)$.



IR(X, Y)	Notation	X
Before	1	Y
Meet	2	Y
Started-by	3	Y
Finished-by	4	Y
Contain	5	Y
Overlapped-by	6	Y
Equal	7	Y
Overlap	8	Y
During	9	Y
Finish	10	Y
Start	11	Y
Met-by	12	Y
After	13	Y
Don't care	0	Y ?

Figure 1：Fourteen relationships of two 1-D intervals.

To illustrate the relationship between two 1-D intervals, besides the thirteen relationships described in Allen's representation, we add one more relationship "don't care", which means that we don't care the relationship between both 1-D intervals. Figure 1 illustrates the fourteen relationships between two 1-D intervals and their notations. Let R be the universal set of the fourteen relationships, where R = {0, 1, 2, …, 12, 13}.

Definition 3. (Multi-dimensional interval) An multi-dimensional interval (MI for short) is denoted as $[I_1, I_2, \ldots, I_n]$, where I_i is the ith dimensional interval, and n is the number of dimensions.

Definition 4. (Multi-dimensional relationship) The multi-relationship (MR for short) between two MIs, $A = [I_{1,1}, I_{1,2}, \ldots, I_{1,n}]$ and $B = [I_{2,1}, I_{2,2}, \ldots, I_{2,n}]$, is denoted as $MR(A, B) = (r_1, r_2, \ldots, r_n)$, where r_i is an 1-D relationship between $I_{1,i}$ and $I_{2,i}$, i.e. $r_i = IR(I_{1,i}, I_{2,i})$, i=1, 2, …, n.

Definition 5. MR_1 contains MR_2, denoted as $MR_2 \subseteq MR_1$, if $r_{1,k} = r_{2,k}$ or $r_{2,k} = 0$, for all $1 \leqq k \leqq n$, where $MR_1 = (r_{1,1}, r_{1,2}, \ldots, r_{1,n})$, and $MR_2 = (r_{2,1}, r_{2,2}, \ldots, r_{2,n})$.

Definition 6. (Pattern) $(MI_1, MI_2, \ldots, MI_k, MR_1, MR_2, \ldots, MR_{k(k-1)/2})$ is a multi-dimensional interval pattern (pattern for short), where MI_i is a multi-dimensional intervals, MR_j is a multi-relationship, i = 1, 2, …, k, j = 1, 2, …, $k(k-1)/2$. The length of a pattern is defined as the number of MIs in the pattern. A pattern of length k is called a *k*-pattern. Note that the MR for eachpair of MIs is recordedin the pattern, and there is no MR for 1-pattern.

Example 1. Consider a pattern (*A*, *B*, *C*, (1, 5), (2, 7), (0, 0)), where *A*, *B* and *C* are MIs, the MR between *A* and *B* is (1, 5), the MR between *A* and *C* is (2, 7), and the MR between *B* and *C* is (0, 0). That is, the 1-D relationship between *A* and *B* in the first dimension is "before", or we can say *A* before *B* in the first dimension. The 1-D relationship between *A* and *B* in the second dimension is "contain", or we can say *A* contains *B* in the second dimension.

Definition 7. (Contain) A pattern $P_1 = (MI_{1,1}, MI_{1,2}, \ldots, MI_{1,k}, MR_{1,1}, MR_{1,2}, \ldots, MR_{1,k(k-1)/2})$ is a sub-pattern of a pattern $P_2 = (MI_{2,1}, MI_{2,2}, \ldots, MI_{2,m}, MR_{2,1}, MR_{2,2}, \ldots, MR_{2,m(m-1)/2})$ if every MI in P_1 can be found in P_2, and the MR between any two MIs in P_1 is contained by the MR of the corresponding MIs in P_2. We can also say that P_2 is a super-pattern of P_1, or P_2 contains P_1.

Example 2. Consider the following two patterns, P_1=(*A*, *B*, (1, 5)) and P_2=(*A*, *B*, *C*, (1, 5), (2, 7), (0, 0)). P_2 is a super-pattern of P_1 since *A* and *B* appear in both $P_1$and P_2, and the MR between *A* and *B* in P_1 is same as that in P_2.

Definition 8. (Frequent) A pattern is *frequent* if its support is not less than *minsup*, where the support of a pattern is defined as the number of transactions containing the pattern in the database, and *minsup* is a user-specified minimum support threshold.

Definition 9. (Closed) A frequent pattern *P* is *closed* if there does not exist any super-pattern of *P* with the same support.

Definition 10. (Projection) The projection of a pattern *P* in a transaction *T* is defined as the positions of the MIs in *T* so that the pattern formed by these MIs contains *P*, denoted as, $T<Pt_1, Pt_2, \ldots, Pt_k>$, where Pt_i is the position of the i^{th} MI in *T*, *i*=1, 2, …, *k*.

Example 3. Consider the MI database in Table 1 and a pattern *P* = (*A*, *C*, (2, 7)). The projection of *P* in T_2 is denoted as $T_2<1, 3>$ since the pattern formed by the first and third MIs in T_2 contains *P*.

Table 1：An example database.

T_1	(A, B, (1, 5))
T_2	(A, B, C, (1, 5), (2, 7), (0, 0))
T_3	(A, B, C, (1, 5), (2, 7), (0, 0))

Definition 11. (Projected Database) The projected database of a pattern *P*, denoted as *PDB*(*P*), contains all the projections of *P* in the database.

The objective of the proposed method is to find all frequent closed multidimensional patterns in a database with respect to the user-specified minimum support threshold.

3. THE PROPOSED METHOD

In this section, we propose an algorithm, called MIAMI (**M**ulti-dimensional **I**nterval P**a**tterns **MI**ne), to mine all frequent closed multi-dimensional patterns from a database. The MIAMI algorithm grows patterns in a depth-first search (DFS) manner. In the mining process, we devise three effective pruning strategies to prune impossible candidates and perform a closure checking scheme to remove non-closed patterns.

3.1 Frequent pattern enumeration

To enumerate frequent patterns, we first list all seed patterns in the database and extend each seed pattern to longer frequent ones in a DFS manner. For each pattern *P*, we generate its frequent super-patterns in two phases. First, we increase the number of NDCs between the MIs in *P*. A 1-D relationship not equal to "don't care" is called a NDC. We call this "dimension growth". Second, we grow the pattern *P* by increasing the number of MIs in the pattern, and call it "MI growth."

Definition 12. (Seed pattern) A seed pattern is a frequent 2-pattern, where only one 1-D relationship between two MIs in the seed pattern is a NDC.

Example 4. Consider the database shown in Table1. Assume *minsup*=2. The seed patterns are (*A*, *B*, (1, 0)), (*A*, *B*, (0, 5)), (*A*, *C*, (2, 0)) and (*A*, *C*, (0, 7)).

3.1.1 Frequent pattern tree

We use a frequent pattern tree to enumerate all frequent patterns, where each node represents a pattern and each child node is a super-pattern of its parent node. The root of the frequent pattern tree is an empty set. We first generate all seed patterns in the database and put them as child nodes of the root node.

Every node in the tree may have two kinds of child nodes. One is generated by the dimension growth whereas the other is generated by the MI growth. Let us consider the database in Table 1. Assume *minsup*=2. All frequent patterns can be enumerated in the frequent pattern tree as shown in Figure 2, where the solid and dotted arrows denote the child patterns grown from the parent pattern by the dimension and MI growth, respectively, and the outer parentheses of each pattern are omitted.

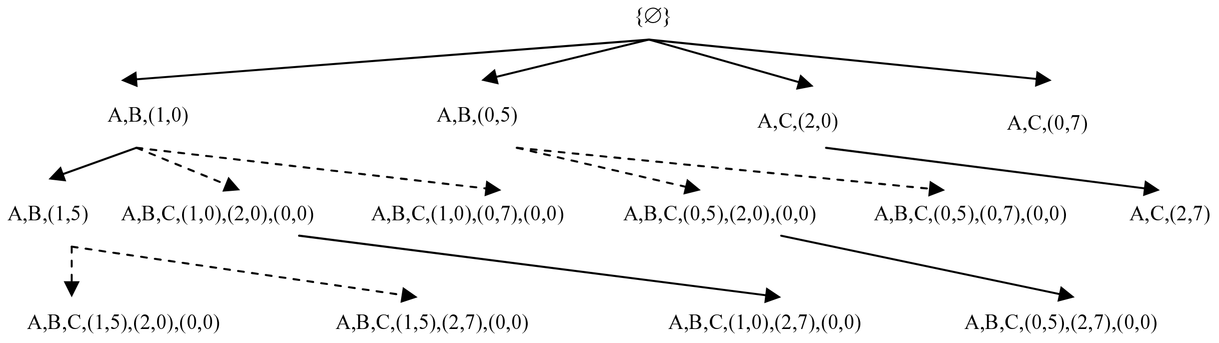


Figure 2：The pattern tree of the database shown in Table 1.

3.1.2 Frequent pattern generation

To generate the super-patterns of a frequent pattern *P*, we first find the seed patterns in *P*'s projected database, and then generate the frequent super-patterns of *P* by joining *P* to each seed pattern joinable to *P*.

Definition 13. (Joinable class) A seed pattern *S* is joinable to *P* if either following condition is satisfied.

1. Dimension growth: *P* contains both MIs in *S*, and the 1-D relationship in a certain dimension in *S* is a NDC, while the corresponding relationship in *P* is "don't care". The pattern of joining *P* to *S* is obtained by replacing the corresponding relationship in *P* with the NDC in *S*.
2. MI growth: *P* and *S* share only one MI in common. The pattern of joining *P* to *S* is *P*', where the MIs and MRs of *P*' are respectively the union of the MIs and MRs of both *P* and *S*, and the remaining unknown MRs of *P*' are marked as "don't care."

Example 5. Consider 3 patterns $P_1 = (A, B, C, (1, 0), (2, 0), (0, 0))$, $P_2 = (A, B, (0, 5))$, and $P_3 = (A, C, (2, 0))$. P_1 is joinable to P_2 since it satisfies the first condition. Joining P_1 to P_2, we have a pattern $P_4 = (A, B, C, (1, 5), (2, 0), (0, 0))$, where the relationship (5) is obtaining by replacing the corresponding relationship of P_1 with the NDC of P_2. P_2 is joinable to P_3 since it satisfies the second condition. Joining P_2 to P_3, we get a pattern

P_5 = (A, B, C, (0, 5), (2, 0), (0, 0)), where both MIs and MRs of P_5 are the union of those in P_2 and P_3. The remaining unknown MR between B and C is marked as "don't care." However, P_3 is not joinable to P_1 because it does not meet any condition.

3.2 Pruning strategies and closure checking

In this section, we devise three pruning strategies and a closure checking scheme to prune impossible candidates and non-closed patterns, where the first two pruning strategies are discussed in Subsection 3.2.1 and the last one is described in Subsection 3.2.2.

3.2.1 Pruning strategies

Based on CHARM (Zaki & Hsiao 2005), we design two pruning strategies to reduce search space while joining a pattern P_1 and a seed pattern S to generate pattern P_2, where S is a frequent seed pattern in $PDB(P_1)$.

1. If $PDB(P_1) = PDB(S)$, then $PDB(P_2) = PDB(P_1) = PDB(S)$. P_1 is not closed since P_2 is a super-pattern of P_1 and both have an identical projected database. Every pattern grown from P_1 is a sub-pattern of a pattern grown from P_2. Thus, we replace every occurrence of P_1 with P_2.
2. If $PDB(P_1) \supset PDB(S)$, then $PDB(P_2) = PDB(S) \subset PDB(P_1)$. Thus, we keep P_1 unchanged and put P_2 as a child node of P_1.

3.2.2 Relationship support checking

The relationship support checking is based on the characteristics of relationship "don't care". That is, the support of relationship "don't care" is equal to the summation of the support of the other 13 relationships. Consider two patterns P_1 and P_2, where all MIs in P_1 and P_2 are the same, and P_1 has one 1-D relationship different from P_2, where the different relationship is "don't care" in P_1 and a NDC in P_2. If P_1 is infrequent, P_2 is infrequent too since the support of P_2 is not greater than that of P_1. By this property, we prune P_2 and skip the mining procedure for P_2.

Example 6. Consider two patterns P_1 = (A, B, C, (1, 1), (13, 1), (0, 0)) and P_2 = (A, B, C, (1, 1), (13, 1), (13, 0)). The only difference between P_1 and P_2 is the relationship between B and C in the first dimension. If P_1's support is less than *minsup*, P_2's supportmust be less than *minsup*. Therefore, we can prune P_2 if P_1 is infrequent.

3.2.3 Closure checking

For a newly generated pattern *P*, we first remove all sub-patterns of *P* that have the same support as *P* in the closed pattern pool. Next, we check if there exists a super-pattern of *P* that has the same support as *P*. If this is not the case, *P* is added to the closed pattern pool.

3.3 The MIAMI algorithm

In this section, wepropose an algorithm, called MIAMI, to mine closed multi-dimensional interval patterns in a DFS manner. The MIAMI algorithm is shown in Figure 3. It contains two procedures, namely, Dimension-Growth and MI-Growth. They are executed subsequently and recursively, and depicted in Figure 4.

```
Algorithm: MIAMI
Input: A database $D$, a user-specified minimum support threshold $minsup$.
Output: All frequent closed patterns $CP$.
1 Scan database $D$ once to find all seed patterns, and collect them into $\Psi$ ;
2 $CP = \varnothing$;
3 Foreach $P$ in $\Psi$. do
4 Check if there exists a super-pattern of $P$ that has the same projected database as $P$; If this is
  the case, prune $P$;
5 Call Dimension-Growth($P$, $minsup$, $CP$);
6 CallMI-Growth($P$, $minsup$, $CP$);
7 Check if $P$ is closed. If this is the case, add $P$ to $CP$ and remove non-closed patterns from
  $CP$;
8 end for
9 Return $CP$;
```

Figure 3：The MIAMI algorithm.

```
Procedure: Dimension-Growth
Input:A pattern $R$, a user-specified minimum support threshold $minsup$.
Output: All frequent closed patterns $CP$.
1 Find all seed patterns in the projected database of $R$ by the pruning strategy in Subsection
  3.2.2 and collect them into $\Psi$ ';
2 $G = \varnothing$;
3 ForeachU in $\Psi$ ' that satisfies the first condition in definition 13 do
4 Apply the pruning strategy described in Subsection 3.2.2 to prune impossible candidate
  patterns;
```

```
5  Let $V = R$ join $U$;
6  Check if $V$ is frequent. Prune $V$ if it is infrequent;
7  Apply the pruning strategies described in Subsection 3.2.1 to update the patterns in $\Psi$'and
   add all frequent patterns found to $G$;
8  end for
9  for each $W$ in $G$ do
10 Check if there exist a super-pattern of $W$ that has the same support and occurrence as $W$. If
   this is the case, prune $W$;
11 Call Dimension-Growth($W$, minsup, $CP$);
12 CallMI-Growth($W$, minsup, $CP$);
13 end for
14 Check if $R$ is closed. If this is the case, add $R$ to $CP$ and remove non-closed patterns from
   $CP$;
15 Return $CP$;
```

Figure 4：The Dimension-Growth procedure.

In the MIAMI algorithm, we first scan the database to find all seed patterns and collect them into Ψ. in step 1. In step 4, we check if there exists a super-pattern of *P* that has the same projected database as *P*. If this is the case, we prune *P*. In steps 5-6, we call the Dimension-Growth and MI-Growth procedures to find the frequent super-patterns of *P* in a DFS manner. In step 7, we check if *P* is a closed pattern. If this is the case, we add *P* to *CP* and remove non-closed patterns from *CP*.

In the Dimension-Growth procedure, we find all seed patterns in the projected database of *P* and collect them into Ψ' with the relationship support checking. In steps 3-8, for each *U* that satisfies the first condition in definition 13, we first apply the pruning strategy described in Subsections 3.2.2 to prune impossible candidates in step 4. Next, we join *R* to *U* to generate a new pattern *V* in step 5. Then, we check if *V* is frequent in step 6. Finally, we apply the pruning strategies described in Subsection 3.2.1 to update the patterns in Ψ' and add all frequent patterns found to *G*. For each pattern *W* in *G*, we check if there is a super-pattern of *W* that has same support and occurrenceas *W*. If this is the case, we prune *W* in step 10. In steps 11-12, we call the Dimension-Growth and MI-Growth procedures to grow the frequent super-patterns of *W*, respectively. In step 14, we check if *R* is closed. If this is the case, we add *R* to *CP* and remove non-closed patterns from *CP*.

The MI-Growth procedureis omitted here because they slightly differ only in steps 3-8. The process in the MI-Growth procedure from steps 3-8 is executed for each *U* in Ψ' satisfies the second condition in definition 13. In short, the Dimension-Growth procedure first grows thedimensions of patterns by increasing the number of NDCs, while the MI-Growth procedure increases the number of MIs.

3.4 An example

Let consider the database in Table 2. Assume that the minimum support threshold is 2. First, we enumerate all seed patterns from the database, namely, P_1 = (*A*, *B*, (1, 0)), P_2 = (*A*, *B*, (0, 5)), P_3 = (*A*, *D*, (2, 0)), P_4 = (*B*, *C*, (8, 0)), P_5 = (*B*, *C*, (0, 9)), and P_6 = (*B*, *D*, (8, 0)). To reduce the candidates of joinable seed patterns to P_1, the pruning strategy in Subsection 3.2.2 is firstly applied to prune impossible seed patterns. Since pattern *Q*=(*A*, *B*, *C* (1, 0), (0, 0), (0, 0)) is infrequent, all the sub-patterns of *Q* are infrequent (relationship support checking). Thus, the seed patterns (*B*, *C*, (8, 0)) and (*B*, *C*, (0, 9)) are removed from the joinable class of P_1. That is, the joinable class of P_1 contains (*A*, *B*, (0, 5)), (*A*, *D*, (2, 0)) and (*B*, *D*, (8, 0)).

Table 2：Another example database.

T_1	(A, B, D, (1, 5), (2, 0), (8, 7))
T_2	(A, B, C, (1, 5), (2, 0), (8, 9))
T_3	(A, B, D, (1, 5), (2, 0), (8, 10))
T_4	(A, B, D, (0, 5), (0, 0), (8, 3))
T_5	(B, C, (8, 9))
T_6	(B, C, (8, 11))

Next, we join P_1 to (*A*, *B*, (0, 5)) and form a super-pattern (*A*, *B*, (1, 5)) by the dimension growth. During the joining process, we find that the projected database of (*A*, *B*, (0, 5)) is equal to the projected database of P_1, which is the first case of the pruning strategies in Section 3.2.1. Thus, P_1 is replaced by (*A*, *B*, (1, 5)). Similarly, in the MI growth, we join P_1 to the other seed patterns and get two super-patterns, namely, P_7 = (*A*, *B*, *D*, (1, 5), (2, 0), (0, 0)) and P_8= (*A*, *B*, *D*, (1, 5), (0, 0), (8, 0)).

To find the frequent super-patterns of P_7, we find the seed patterns in the projected database of P_7. The seed pattern found is (*B*, *D*, (8, 0)). We join the seed pattern to P_7 and find that this is also the first case of the pruning strategies in Section 3.2.1. Thus, we replace P_7 by the newly generated super-pattern (*A*, *B*, *D*, (1, 5), (2, 0), (8, 0)). Since there are not available seed patterns in P_7's projected database, we use the closure checking scheme to check whether P_7 is closed or not. Because the closed pattern pool is empty, P_7 is added to the closed pattern pool. We then continue to P_8 and find that P_8 is a sub-pattern of P_7 and both have the same support. Thus, P_8 is removed. We backtrack to P_1 and check if P_1 is closed. Although P_1 is a sub-pattern of P_7, they have different supports. Thus, P_1 is added to the closed pattern pool.

Now, we continue to P_2. Note that although P_2 is contained by P_7, they have different supports. We generate two super-patterns of P_2, namely, $P_9 = (A, B, D, (0, 5), (2, 0), (0, 0))$ and $P_{10} = (A, B, D, (0, 5), (0, 0), (8, 0))$. Since the projected databases of the seed patterns $(A, D, (2, 0))$ and $(B, D, (8, 0))$ are contained by P_2's projected database, this is the second case of the pruning strategies in Section 3.2.1. We put P_9 and P_{10} as a child node of P_2. However, P_9 is not closed and thusremoved. P_{10} is added to the pool since it satisfies closure checking scheme. Similarly, we repeat the same procedure and find that both $(B, C, (8, 0))$ and $(B, C, (8, 9))$ also satisfy closure checking scheme, and thus, are added to the closed pattern pool, too. P_5 and P_6 are non-closed patterns and removed subsequently.

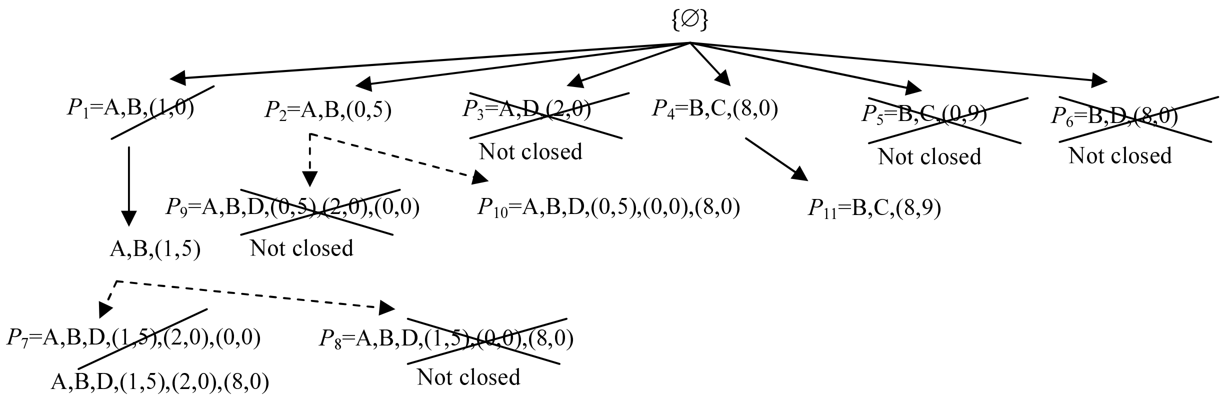


Figure 5：The mining process for the database shown in Table 2.

Finally, the MIAMI algorithm finds six closed patterns $P_1=(A, B, (1, 5))$, $P_2 = (A, B, (0, 5))$, $P_4= (B, C, (8, 0))$, $P_7=(A, B, D, (1, 5),(2, 0),(8, 0))$, $P_{10} = (A, B, D, (0, 5),(0, 0),(8, 0))$ and $P_{11} = (B, C, (8, 9))$ as shown in Figure 5, where the patterns marked by X mean that they are removed by closure checking scheme and the patterns marked by a slash means that they are replaced by the pruning strategies in Subsection 3.2.1.

4. PERFORMANCEEVALUATION

In this section, we conducted the experiments to evaluate the performance of the MIAMI algorithm and the modified Apriori algorithm by using both synthetic and real datasets. Both algorithms were implemented by C++, Microsoft Visual Studio 2008. All of experiments were performed on an IBM compatible PC with Intel Core 2 Quad CPU Q9400 @ 2.66GHz, 2GB main memory, running on Windows XP Professional.

The modified Apriori algorithm (Agrawal & Srikant1994) grows patterns in a breath-first search manner. It first finds all seed patterns from the database. Next, itgenerates candidate 2-patterns by growing the dimensions for each seed pattern, which is similar to the Dimension-Growth procedure in the MIAMI algorithm. Then, it scans the database to count the support for each 2-pattern generated and finds all closed 2-patterns satisfying closure checking. Subsequently, it joins the closed 2-patterns to each joinable seed pattern and generates all candidate 3-patterns. Then, it scans the database once to find the support of each candidate, and checks if each candidate satisfies frequency and closure requirement, which is similar to the MI-Growth procedure in the MIAMI algorithm. Similarly, itrepeatedly uses closed (k-1)-patterns to generate closed k-patterns through extending both dimension and MI of patterns until no more closed patterns can be generated.

4.1 Synthetic data

The synthetic data generator is similar to the one used in Agrawal and Srinkant (1995) with some modifications, where a transaction contains a sequence of MIsandthe MRs between those MIs. First, we generate potential patterns whose lengths follow a Poisson distribution with mean equal to λ_1 Then, we randomly select a pattern from the potential patterns and use it to form a transaction whose length also follow a Poisson distribution with mean equal to λ_2 Table 3 lists the parameters and the default settings used in the synthetic data generator. Note that the support of a pattern is defined as the fraction of transactions containing the pattern in the database in the experimental section.

Table 3：Parameters used to generate synthetic data.

Meaning	Setting
Number of potential patterns	500
Average length of potential patterns	5
Average length of transactions	5
Number of MIs	20
Number of dimensions	3
Number of transactions	50K
Minimum support	0.4%

4.2 Performance evaluation on synthetic data

In this section, we compared the MIAMI algorithm and the modified Apriori algorithm by varying one parameter and keeping the others at the default values as shown in Table 3. Figure 6 shows the runtime versus the minimum support threshold, where the minimum support threshold varies from 0.3% to 0.5%. The MIAMI algorithm runs 26%-67% faster than the modified Apriori algorithm, while the modified Apriori algorithm runs out of main memory due to generating a large number of candidates when the minimum support threshold is 0.3%. Since the MIAMI algorithm generates patterns in a DFS manner, and employs the projected databases to localize the support counting and pattern joins. Moreover, it employs three effective pruning strategies to reduce search space and performs a closure checking scheme to remove non-closed patterns. Therefore, the runtime of the MIAMI algorithm increases slower than that of the modified Apriori algorithm when the minimum support threshold decreases.

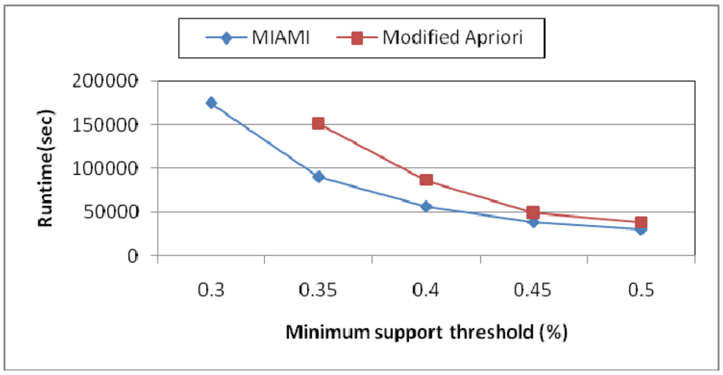


Figure 6：Runtime versus minimum support.

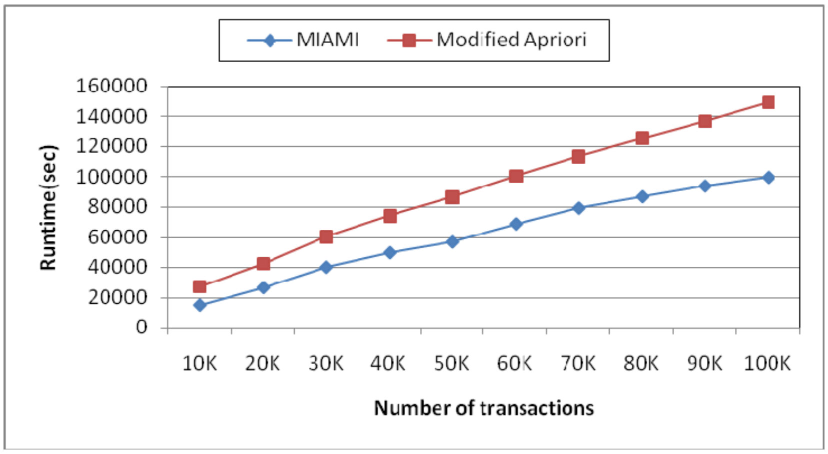


Figure 7：Runtime versus number of transactions.

Figure 7 shows the runtime versus the number of transactions for both algorithms, where the number of transactions varies from 10K to 100K. As the number of transactions increases, the runtime of both algorithms increases almost linearly. Since the modified Apriori algorithm generates a large amount of candidates and needs more database scans, the runtime of the modified Apriori algorithm is more than that of the MIAMI algorithm.

Figure 8 shows the runtime versus the number of dimensions for both algorithms, where the number of dimensions varies from 1 to 5. As the number of dimensions increases, the number of closed patterns will increase sharply due to a large number of combinations of different dimensions. Therefore, the runtime of both algorithms increases sharply when the number of dimensions increases. However, the MIAMI algorithm runs 10%-50% faster than the modified Apriori algorithm, because the modified Apriori algorithm mines patterns level by level and generates a huge number of candidates and non-closed patterns, while the MIAMI algorithm can prune many impossible candidates and non-closed patterns.

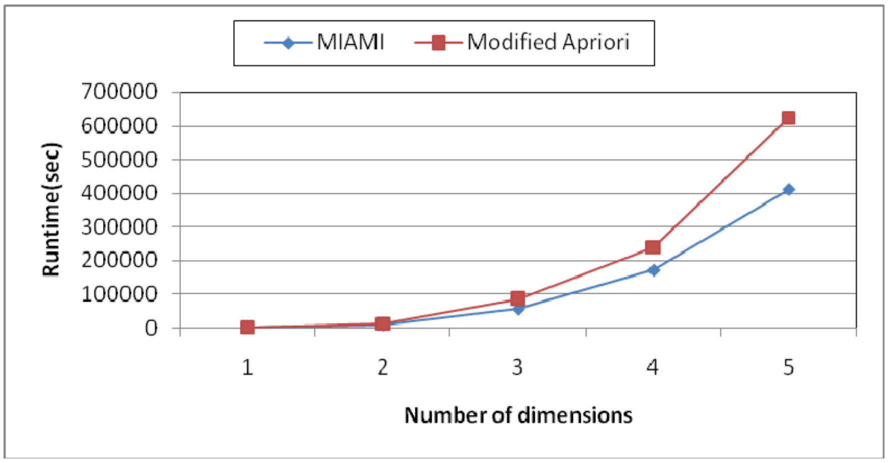


Figure 8：Runtime versus number of dimensions.

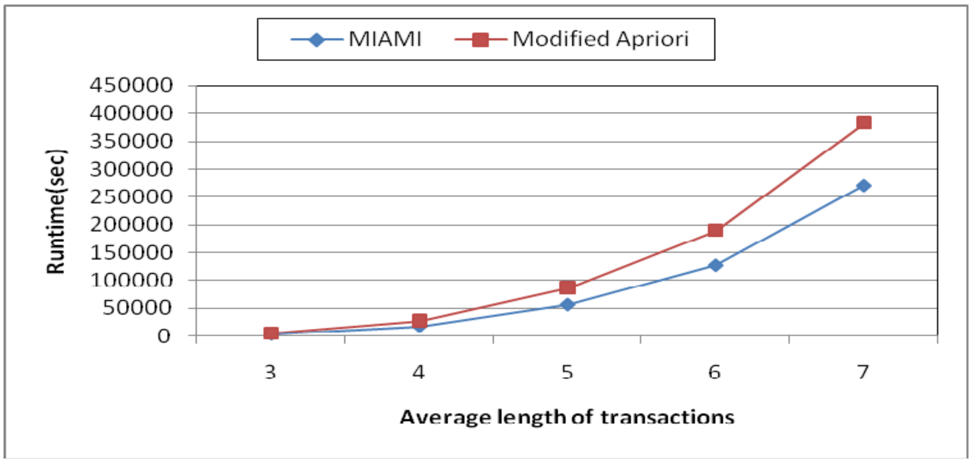


Figure 9：Runtime versus average length of transactions.

Figure 9 shows the runtime versus the average length of transactions, where the average length varies from 3 to 7. The number of closed patterns increases dramatically as the average length of transactions increases. Thus, the runtime of both algorithms increases quickly as the average length of transactions increases. The runtime of the MIAMI algorithm runs 40% to 52% faster than the modified Apriori algorithm since the modified Apriori algorithm generates too many non-closed patterns in each level.

Figure 10 shows the runtime versus the number of MIs for both algorithms, where the number of MIs varies from 20 to 40. The runtime of both algorithms decreases sharply while the number of MIs increases. The MIAMI algorithm runs 20%-50% faster than the modified Apriori algorithm because it generates much fewer candidates and non-closed patterns. The modified Apriori algorithm generates fewer candidates as the number of MIs increases because more impossible candidates are pruned. Thus, the runtime of the modified Apriori is closed to the runtime of the MIAMI algorithm while the number of MIs is large.

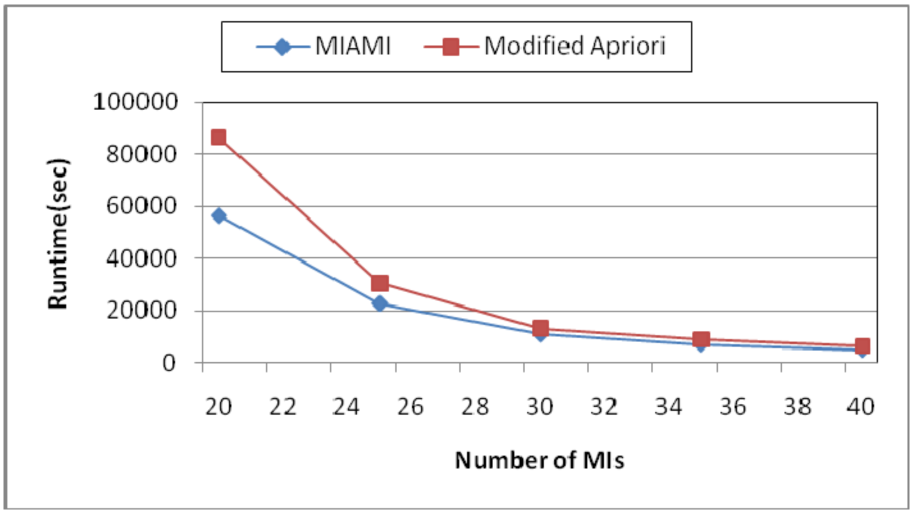


Figure 10：Runtime versus number of MIs.

4.3 Performance evaluation on real data

We used a real dataset to evaluate the performance of the proposed method. The dataset was collected from Yahoo! Finance (http://finance.yahoo.com/) from August 2005 to March 2011. We selected three firms in the telecommunication industry sector, namely Chunghwa Telecom (CHT), Taiwan Mobile (MYFONE), and Far Eastone Telecom (FET).

Every transaction in the dataset is formed weekly and contains three MIs, namely CHT, MYFONE, and FET, where each MI has four dimensions. Each dimension records the interval of a certain value in a week: the first dimension-close prices, the second dimension-trading volumes, the third dimension-price fluctuation (defined as the ratio of the daily highest price minus the lowest price to the close price), and the fourth dimension-return of investments (ROI defined as the ratio of money gained or lost to the amount of money purchased in the previous day).

The performance of both algorithms using the real dataset is quite similar to that found using the synthetic data. Figure 11 shows the runtime versus the minimum support threshold for both algorithms, where the minimum support threshold varies from 14% to 24%. As the minimum support decreases, the MIAMI algorithm outperforms the modified Apriori algorithm which generates numerous candidates and non-closed patterns. Thus, MIAMI algorithm is much more efficient and scalable than the modified Apriori algorithm.

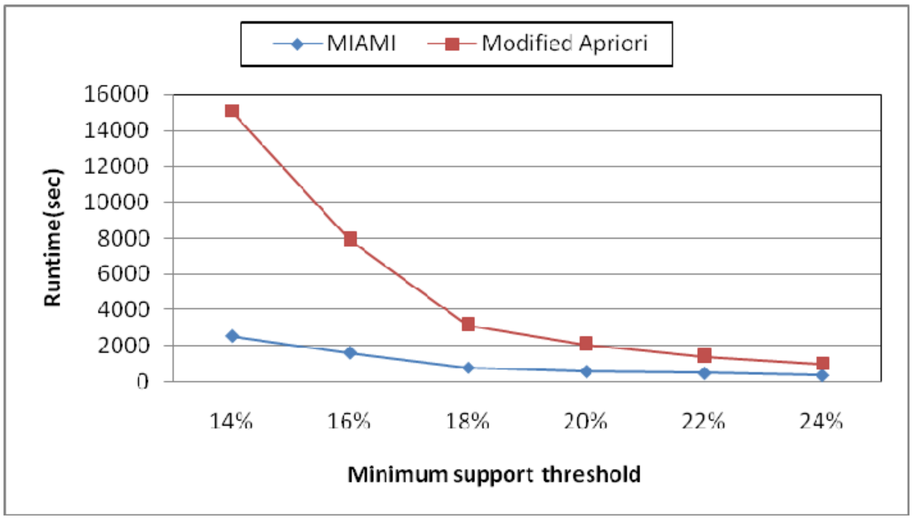


Figure 11：Runtime versus minimum support threshold.

In this real dataset, some interesting patterns would be found. For example, pattern P_1 = (CHT, MYFONE, FET, (13, 13, 0, 0), (13, 13, 0, 0), (13, 0, 0, 0)) shows that CHT is usually greater than the others in both stock price and trading volume and holds the leading position in the telecommunication industrysector. Also, pattern P_2 = (CHT, MYFONE, FET, (0, 0, 6, 0), (0, 0, 6, 0), (0, 0, 8, 0)) indicates that CHT usually has the least price fluctuation. However, pattern P_3 = (CHT, MYFONE, FET, (0, 0, 8, 0), (0, 0, 8, 0), (0, 0, 0, 0)) reveals that the price fluctuation of CHT overlaps with those of MYFONE and FET and pattern P_4 = (CHT, MYFONE, FET, (9, 0, 0, 6), (9, 0, 0, 6), (0, 0, 0, 0)) indicates while the price of CHT is contained by those of MYFONE and FET, the ROI of CHT is overlapped bythose of MYFONE and FET. In year 2008, CHT suffered from twice significant price decline due to the great loss to revenue. One was caused by the financial shortfall of 4 billion dollars because of the appreciation of NT dollars and the otherwas from excluding dividend and rightwhich directly diminished the market capitalization by more than 40 billion NT dollars. Excluding dividend and right is to return parts of profits of a company to its shareholders in form of, primarily, cash or stocks. In general, a stock's price drops roughly by the amount of the dividend and rightwhich results from the economic value transfer and also to keep the equivalence of rights and interests for the stock buyers before/on the Excluding Dividend (ED) day. Thus, the price fluctuation of CHT was much greater than those of MYFONE and FET, and the ROI of CHT was also worse than those of MYFONE and FET in year 2008.

In addition, pattern P_5 = (MYFONE, FET, (1, 0, 0, 6)) depicts that the ROI of MYFONE is overlapped by that of FET only when the price of FET is greater than thatof MYFONE. In year 1998, FET was the first company that launched a fully integrated dual-band (GSM 900 and GSM 1800) system in the world so that FET provided better quality services. Also, FET proposed many innovative marketing strategies, such as, pre-paid cards and charge in seconds. Moreover, FET had numerous and unique channels of distributions. Thus, it has higher stock price and ROI than MYFONE at that time. However, another pattern P_6 = (MYFONE, FET, (5, 8, 8, 0)) shows that the volume and price fluctuation of MYFONE are overlapped with that of FET when the price of MYFONE contains that of FET. Around years 2008-2009, MYFONE completed a series of mergers and acquisitions with three companies, including TransAsia Telecommunication, Mobitai Communications and Taiwan Fixed Network. Thus, MYFONE became the pioneering company offering "quadruple play" services namely, mobile, fixed-line, cable TV, and broadband. MYFONE launched three brands namely, Taiwan Mobile, TWM Broadband, and TWM Solution, to promote its quadruple play services for the consumer, household and enterprise markets, and thussubstantially increased the coverage of services and users. From both patterns, we observe that FET aggressively provided good services for users and had higher stock price and ROI than MYFONE in the early stage; however, MYFONE outperformed FET after a series of mergers and acquisitions in years 2008-2009.

In summary, MIAMI algorithm generates patterns in a DFS manner and employs the projected databases to localize support counting and pattern joins. Moreover, it employs three effective pruning strategies to reduce search space and performs a closure checking scheme to remove non-closed patterns. Therefore, MIAMI algorithm is more efficient and scalable than the modified Apriori algorithm in both synthetic and real datasets.

5. CONCLUSIONS AND FUTURE WORK

We have proposed an efficient algorithm, called MIAMI, to mine closed multi-dimensional interval patterns in a database. The MIAMI algorithm first finds all seed patterns in the database, and then grows closed patterns in two phases, namely, Dimension-growth and MI-growth. We grow a frequent pattern to find longer frequent ones by increasing the number of NDCs between the MIs in the pattern in the dimension growth phase and increasing the number of MIs in the pattern in the MI growth phase. During the mining process, we employ three effective pruning strategies to eliminate impossible candidates and a closure checking scheme to remove non-closed patterns. Thus, the proposed algorithm can efficiently mine closed multi-dimensional interval patterns in databases. The experimental results show that the proposed algorithm is more efficient and scalable than the modified Apriori algorithm in the synthetic and real datasets.

By introducing the "don't care" relationship, we can consider every possible combination of M-D interval patterns. In addition, if each MI in a pattern represents 1-D interval, the pattern will become a 1-D interval pattern. Thus, mining 1-D interval patterns is a special case of mining M-D interval patterns. That is, we generalize 1-D into M-D interval patterns. Therefore, we are able to mine patterns in a more flexible way and discover more interesting patterns.

Our proposed algorithm MIAMI can be applied in many ways wherever the applications are realized by multi-dimensional intervals. For example, the investors may raise funds on the companies in stock market from the investigation of some interval indices, such as, stock price, trading volume, and volume of sales in marketplace. Managers might wonder the factors which contribute to better productivity by observing the relationships among human forces, the probability of malfunctioning, the satisfaction of labors, and the level of automation. Home buyers may investigate the trends of real estate if the range of prices and geographic zones of houses and apartments are recorded. The government may work on the primary weaknesses of administration by identifying those lagging indices, such as, the rate of unemployment, GDP, national income, and the rate of economic growth. Or, the government may make policies according to the comprehension of the gender equivalence through the analysis of the varied relationships in terms of both ages and average salaries. Thus, the MIAMI algorithm significantly broadens the applications as well as enriches the managerial implications and decisions.

Although the MIAMI algorithm can efficiently mine closed multi-dimensional interval patterns, it can be further improved to address other issues. It is worth extending the method further to mine closed patterns with some constraints to mine patterns in need and exclude the unnecessary patterns. In addition, it may be meaningful to consider the relationships between different dimensions in real world applications. Thus, the MIAMI algorithm can be extended to mine the patterns with the relationship between different dimensions. Moreover, it is helpful to design a parallel algorithm to accelerate the mining process in the future.

Acknowledgements

The authors are grateful to the anonymous referees for their helpful comments and suggestions. This research was supported in part by the National Science Council of Republic of China under Grant No. NSC 100-2410-H-002-023-MY3.

References

Agrawal, R. and Srikant, R. (1994),'Fast algorithms for mining association rules', *Proceedings of the International Conference on Very Large Data Bases*, Vol. 1215, pp. 487-499.

Agrawal, R. andSrikant, R. (1995), 'Mining sequential patterns', *Proceedings of the International Conference on Data Engineering*, Taipei, Taiwan, pp. 3-14.

Allen, J.F. (1983),'Maintaining knowledge about temporal intervals', *Communications of the ACM*, Vol. 26, No.11, pp.832-843.

Amo, S. de, Junior, W.P. andGiacometti, A. (2008), 'MILPRIT*: A constraint-based algorithm for mining temporal relational patterns', *International Journal of Data Warehousing and Mining*, Vol. 4, No. 4, pp. 42-61.

Guyet, T.and Quiniou, R. (2008), 'Mining temporal patterns with quantitative intervals', *Proceedings of International Conference on Data Mining*, pp. 218-227.

Han, J., Pei, J. and Yin, Y. (2000),'Mining frequent patterns without candidate generation', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Vol. 29, No.2, pp. 1-12.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. and Hsu, M.C. (2000),'FreeSpan: Frequent pattern-projected sequential pattern mining', *Proceedings of International Conference on Knowledge Discovery and Data Mining*, Boston, USA, pp.355-359.

Höppner, F. (2001),'Learning temporal rules from state sequences', Proceedings of International Joint Conferences on Artificial Intelligence Workshop on Learning from Temporal and Spatial Data, Seattle, USA, pp. 25-31.

Kam, P.-S. and Fu, A. W.-C. (2000), 'Discovering temporal patterns for interval-basedevents', *Proceedings of Second International Conference on Data Warehousingand Knowledge Discovery*, London, UK, pp. 317-326.

Kong, X., Wei, Q. and Chen, G. (2010), 'An approach to discovering multi-temporal patterns and its application to financial databases', *Information Sciences*, Vol. 180, No. 6, pp. 873-885.

Lee, Y.J., Lee, J.W., Chai, D.J., Hwang, B.H. and Ryu, K.H. (2009), 'Mining temporal interval relational rules from temporal data', *The Journal of Systems and Software*, Vol. 82, No. 1, pp. 155-167.

Papapetrou, P., Kollios, G., Sclaroff, S. and Gunopulos, D. (2009),'Mining frequent arrangements of temporal intervals', *Knowledge and Information Systems*, Vol. 21, No. 2, pp. 133-171.

Patel, D., Hsu, W. and Lee, M. (2008), 'Mining relationships among interval-based events for classification', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada, pp. 393-404.

Pei, J., Han, J. and Mao, R. (2000), 'CLOSET: An efficient algorithm for mining frequent closed itemsets', *Proceedings of the 5th ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, USA, pp.11-20.

Pei, J., Han, J., Mortazavi-Asl, B. and Pinto, H. (2001), 'PrefixSpan：Mining sequential patterns efficiently by prefix-projected pattern growth', *Proceedings of the17th International Conference on Data Engineering*, *Heidelberg*, Germany, pp.215-224.

Winarko, E. and Roddick, J.F. (2007), 'ARMADA - An algorithm for discovering richer relative temporal association rules from interval-based data', *Data and Knowledge Engineering*, Vol. 63, No. 1, pp. 76-90.

Wu, S.-Y. and Chen, Y.-L. (2007),'Mining nonambiguoustemporalpatterns for interval-based events', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 6, pp. 742-758.

Wu, S.-Y. and Chen, Y.-L. (2009), 'Discovering hybrid temporal patterns from sequences consisting of point- and interval-based events', *Data and Knowledge Engineering*, Vol. 68, No. 11, pp. 1309-1330.

Zaki, M.J. (2001), 'SPADE: An efficient algorithm for mining frequent sequences', *Machine Learning*, Vol. 42, No. 1-2, pp.31-60.

Zaki, M.J. and Hsiao, C.-J (2005), 'Efficient algorithms for mining closed itemsets and their lattice structure', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 4, pp. 462-478.