Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Sciences 176 (2006) 1752–1780
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Flexible online association rule mining based on multidimensional pattern relations

Ching-Yao Wang [a,*], Shian-Shyong Tseng [a], Tzung-Pei Hong [b]

[a] *Institute of Computer and Information Science, National Chiao-Tung University, Hsinchu 300, Taiwan, ROC*
[b] *Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan, ROC*

Received 20 April 2004; received in revised form 1 May 2005; accepted 2 May 2005

Abstract

Most incremental mining and online mining algorithms concentrate on finding association rules or patterns consistent with entire current sets of data. Users cannot easily obtain results from only interesting portion of data. This may prevent the usage of mining from online decision support for multidimensional data. To provide ad-hoc, query-driven, and online mining support, we first propose a relation called the *multidimensional pattern relation* to structurally and systematically store context and mining information for later analysis. Each tuple in the relation comes from an inserted dataset in the database. We then develop an online mining approach called *three-phase online association rule mining* (TOARM) based on this proposed multidimensional pattern relation to support online generation of association rules under multidimensional considerations. The TOARM approach consists of three phases during which final sets of

* Corresponding author. Tel.: +886 3 571 2121x56658; fax: +886 3 572 1490.
E-mail addresses: cywang@cis.nctu.edu.tw, simon.cis89g@nctu.edu.tw (C.-Y. Wang), sstseng@cis.nctu.edu.tw (S.-S. Tseng), tphong@nuk.edu.tw (T.-P. Hong).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2005.05.005

patterns satisfying various mining requests are found. It first selects and integrates related mining information in the multidimensional pattern relation, and then if necessary, re-processes itemsets without sufficient information against the underlying datasets. Some implementation considerations for the algorithm are also stated in detail. Experiments on homogeneous and heterogeneous datasets were made and the results show the effectiveness of the proposed approach.

Keywords: Data mining; Association rule; Incremental mining; Multidimensional mining; Constraint-based mining; Data warehouse

1. Introduction

Data mining technology has become increasingly important in the field of large databases and data warehouses. This technology helps discover nontrivial, implicit, previously unknown, and potentially useful knowledge [3,12,19], thus aiding managers in making good decisions. Mining association rules from transaction databases is the most interesting and popular of the various types of databases and mined knowledge. It discovers relationships among items such that the occurrence of certain items in a transaction implies the occurrence of certain other items in the same transaction [2,4].

Previous works on mining association rules can be classified into batch mining [4,5,10,20,27–29,31,34,36] and incremental mining approaches [6,13,14,16,22,33,35] according to their processing methods. Most focus on finding association rules or patterns in specified parts of databases that satisfy specified criteria (such as minimum support and minimum confidence) [7,9,18,26,30]. Some contexts (circumstances) such as region, time, and branch are usually ignored in mining requests. However, decision-makers frequently must consider diverse aspects of problems [17,18]. They may need to analyze market demands, customer preferences, locals, and short-term/long-term trends. They may also want to understand changes in discovered patterns or rules in various dimensions. Some examples are shown below.

Scenario 1. A decision-maker may know what product combinations sold last August were popular, and want to know what product combinations sold last September were also popular.

Scenario 2. A decision-maker may know from a transaction database that people often buy beer and diapers together, and want to know under what circumstances (e.g., place, month, or branch) this pattern is significant or, conversely, under what circumstances this pattern becomes insignificant.

Scenario 3. A decision-maker may want to know how the mined patterns this year differ from those found last year, e.g., what new patterns have appeared and what old patterns have disappeared.

Scenario 4. A marketing analyst may want to analyze all first-quarter data from branches in Los Angeles and San Francisco over the last five years.

Scenario 5. A marketing analyst may want to know what patterns are significant when the minimum support is increased from 5% to 10%.

The examples above all require more context information to describe the problem domain. A mining algorithm that can handle relevant context information in mining requests will thus help decision-makers consider various aspects of problems in diverse ways.

In this paper, we attempt to extend the concept of *effectively utilizing patterns previously discovered* to support online generation of association rules under multidimensional considerations. We first propose the *multidimensional pattern relation* to structurally and systematically store additional context information and mining information for each inserted dataset. Conceptually, a multidimensional pattern relation is similar to a data warehouse for OLAP [11,23,37], except that it is used to store mined patterns but not data. We then develop a *three-phase online association rule mining* (TOARM) approach based on the proposed multidimensional pattern relation to effectively and efficiently satisfy diverse mining requests. It consists of three main phases, *candidate itemset generation*, *candidate itemset reduction*, and *association rule generation*. The candidate itemset generation phase selects tuples that satisfy the context constraints in mining requests and generates candidate itemsets from the matching tuples. The candidate itemset reduction phase then calculates upper-bound supports for the candidate itemsets and uses two pruning strategies to reduce the number of candidates. Finally, the association rule generation phase finds final large itemsets and derives association rules from them. Experimental results show the effectiveness of the proposed TOARM approach.

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. An overview of the proposed methodology is given in Section 3. The multidimensional pattern relation used in this paper is defined in Section 4. Some relevant properties of multidimensional online mining are stated and proven in Section 5. The TOARM algorithm is proposed in Section 6. Implementation considerations for the algorithm are stated in Section 7. Experimental results are presented and discussed in Section 8. Conclusions and future work are given in Section 9.

2. Related work

Mining association rules from transaction databases has recently become one of the most interesting and popular research topics in data mining. An association rule indicates a relationship among items such that the occurrence of certain items in a transaction implies the occurrence of certain other items in the same transaction. The process of mining association rules can be roughly divided into two tasks [4]: *finding large itemsets* and *generating interesting association rules*. The first task discovers itemsets in a given database that satisfy a user-specified *minimum support* criterion. It is used to obtain the statistically significant patterns. The second task finds association rules in the large itemsets that satisfy a user-specified *minimum confidence* criterion. Since the first task is very time-consuming compared to the second one, the major challenges in mining association rules thus focus on how to reduce the search space and decrease the computation time required for the first task. Some famous mining algorithms, such as Apriori [4], DIC [10], DHP [31], Partition [34], Sampling [28], GSP [5], and FP-Growth [20,36], were proposed to achieve this purpose. Among them, the Apriori algorithm, which is the most well known, utilizes a level-wise candidate generation approach to reduce its search space such that only large itemsets found in the previous level are treated as seeds for generating candidate itemsets in the current level. This level-by-level property can greatly reduce the number of itemsets considered in a mining process. Many later algorithms were based on this property and attempted to further reduce candidate itemsets and I/O costs. For example, the Partition algorithm proposed by Savasere et al. reduces I/O costs by partitioning a database into small chunks that can be loaded into main memory [34]. The algorithm scans a database only twice to find large itemsets. It generates the set of all potentially large itemsets in each chunk during the first pass, and then counts their global support during the second pass. Comprehensive overviews can be found in [12,19].

Researchers have recently developed incremental mining algorithms for maintaining association rules without re-processing the entire database whenever it is updated. Examples include the FUP-based algorithms proposed by Cheung et al. [13,14], the adaptive algorithm proposed by Sarda and Srinivas [33], the incremental mining algorithm based on the concept of *pre-large itemsets* proposed by Hong et al. [22], and the incremental updating technique based on the concept of *negative border* proposed by Thomas et al. [35] and Feldman et al. [16]. The common idea among these approaches is that previously mined patterns are stored in advance for later use. When new transactions are inserted or old records are deleted, a large part of the final results can be obtained by comparing the patterns mined from the newly inserted transactions or deleted records with the pre-stored mined knowledge. Only a small portion of the patterns need be re-processed against the entire database, thus saving much computation time. Among the approaches mentioned above,

the FUP-based algorithms [13,14] store the previously mined large itemsets for later maintenance. Some other approaches utilize *pre-large itemsets* [22] and *negative borders* [16,35] to enlarge the amount of pre-stored mined information, further improving maintenance performance at the expense of storage space.

Researchers have also developed online mining algorithms to obtain required sets of patterns without re-processing the entire database whenever user-specified thresholds are changed. Examples are the OLAP-style algorithm proposed by Aggarwal and Yu [1] and the Carma algorithm proposed by Hider [21]. The OLAP-style algorithm is quite similar to a typical incremental mining algorithm that utilizes previously mined patterns to save on I/O and computation. It first stores *primary itemsets* based on a low minimum support criterion in a *latticed* data structure, and then responds to users' queries with higher minimum support criteria by processing the lattice. It thus preprocesses the data just once, but can efficiently handle multiple user queries. The Carma algorithm attempts to provide intermediate results as feedback to users while databases or minimum support thresholds are being changed. Users are thus able to dynamically adjust thresholds according to intermediate results. The Carma algorithm uses two runs. During the first run, it constructs a lattice composed of all potential large itemsets from the transactions. Each itemset in the lattice uses a lower bound and an upper bound to record its possible support range. When a mining request is input, itemsets in the lattice whose support ranges cover or are larger than the new minimum support threshold are output to the second run. During the second run, the Carma algorithm finds the precise support for each itemset from the first run to determine whether it is truly large.

All of the incremental mining and online mining algorithms mentioned above seek sets of association rules or patterns consistent with the entire set of data present at time of search. They do not consider the contexts (circumstances) such as region, time, and branch in mining requests. Users cannot easily obtain association rules or patterns from only interesting portions of data. This may produce additional rules that are irrelevant and uninteresting to users. Constraint-based and multidimensional mining techniques [7,9,17,18,24,26,30,32] which allow users to specify constraints as a guidance have thus been developed to extract interesting knowledge from a data warehouse or a database. For example, Kamber et al. [24] proposed a famous approach that allowed users to specify the predicates that appear in antecedent and consequent parts of association rules. Their approach mined rules directly from data stored in data warehouses. Different from their approach, this paper systematically mines association rules from each incoming dataset and stores the rules with relevant context information in a structural repository for later mining requests. Specifically, the proposed TOARM approach gets the multidimensional association rules mainly from stored patterns, while the previous

multidimensional mining approaches get the rules from data in data warehouses.

Interestingly, many large organizations have multiple databases distributed at different branches. Traditional data mining algorithms may put all data from different databases in a common repository for centralized analysis. This kind of mining causes some problems. The collected data may be too huge to be coped with. Besides, some useful rules or patterns regarding local databases may be lost. As a result, multi-database mining has recently been recognized as an important research topic and some studies [25,38,39] on mining association rules over multi-databases have been proposed. These approaches mined rules or patterns at different databases and gathered the mined results. They did not, however, maintain a repository to systematically and structurally store the mining information and related context information for later flexible analysis. It is thus not easy for them to provide enough online decision support for the scenarios given in Section 1.

3. Overview of the proposed methodology

We first propose a relation called the *multidimensional pattern relation* to structurally and systematically store context information and mining information for later analysis. Each tuple in the relation comes from an inserted dataset in a database. We then develop an online mining approach called *three-phase online association rule mining* (TOARM) based on the proposed multidimensional pattern relation to support online generation of association rules under multidimensional considerations. Although the proposed approach is based on the concept of *effectively utilizing patterns previously discovered*, the TOARM approach with the multidimensional pattern relation is not intended to deal with the incremental problems, but with the multidimensional mining requests.

The multidimensional pattern relation is conceptually similar to a data warehouse for OLAP [11,23,37]. Both preprocess the underlying data in advance, integrate related context information, and store the results in a centralized structural repository for later use and analysis. The multidimensional pattern relation consists of two major types of information for providing adhoc, query-driven, and online mining support. One is context information such as region, time, and branch for diverse decision support, and the other is mining information such as number of transactions and previously mined patterns for efficient online mining. The multidimensional pattern relation is thus used to store mined patterns instead of data.

Assume data is inserted or deleted in a block during a time interval such as a month. Whenever a new block of data is inserted into a database, significant patterns are mined from this dataset based on an initial minimum support and act as the mining information. The mining information along with

corresponding context information is then stored in the pre-defined multidimensional pattern relation. On the other hand, when an old block is deleted from the database, its corresponding context information and mining information are also removed from the multidimensional pattern relation. Fig. 1 shows an example of a database consisting of blocks of data from various branches from 2002/1 to 2002/12. The context information and the mining information along with each block of data form a tuple in the corresponding multidimensional pattern relation. When a new block of data from the Los Angeles branch in 2003/1 is inserted, it is first stored in the underlying database. The significant patterns for this block are mined and then stored with other context information in the multidimensional pattern relation.

Unlike the summarized information on *fact attributes* in a data warehouse, the mined patterns in the multidimensional pattern relation cannot be directly aggregated to satisfy users' mining requests. Assume the minimum support in a mining request is always greater than or equal to that used to obtain the mining information in the multidimensional pattern relation. The proposed TOARM approach consists of three phases to find the final sets of patterns under multidimensional considerations. TOARM first selects and integrates related mining information in the multidimensional pattern relation, and then, if

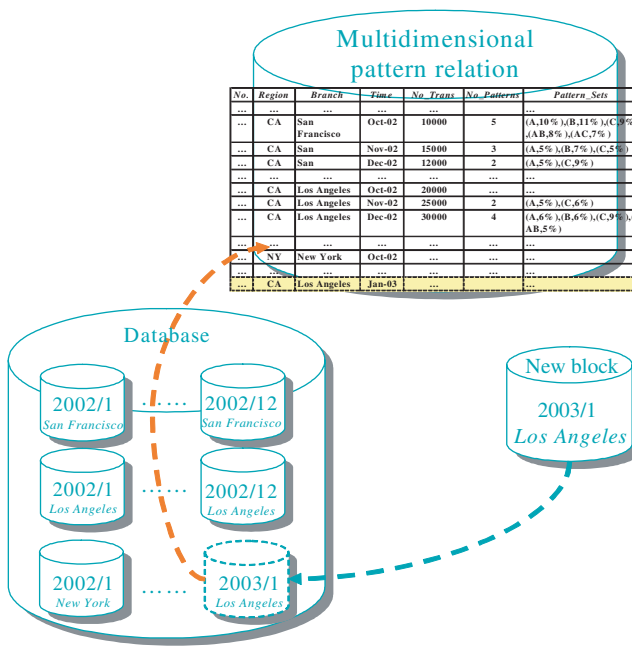

Fig. 1. A database and its corresponding multidimensional pattern relation.

necessary, re-processes itemsets without sufficient information against the underlying datasets.

4. The multidimensional pattern relation

In this section, we formally define the *multidimensional pattern relation* for storing context information and mining information for later analysis. First, a relation schema R, denoted by $R(A_1, A_2, \ldots, A_n)$, is made up of a relation name R and a list of attributes A_1, A_2, ..., A_n. Each attribute A_i is associated with a set of attribute values, called the domain of A_i and denoted by $\mathrm{dom}(A_i)$. A relation r of the relation schema $R(A_1, A_2, \ldots, A_n)$ is a set of tuples $\{t_1, t_2, \ldots, t_m\}$. Each tuple t_i is an ordered list of n values $\langle v_{i1}, v_{i2}, \ldots, v_{in}\rangle$, where each value v_{ij} is an element of $\mathrm{dom}(A_j)$.

A multidimensional pattern relation schema (MPR) is a special relation schema for storing mining information. An MPR consists of three types of attributes: *identification* (ID), *context*, and *content*. There is only one identification attribute for an MPR. It is used to uniquely label tuples. Context attributes describe the contexts (circumstances) of an individual data block, gathered together from a specific business viewpoint. Examples of context attributes are region, time, and branch. Content attributes describe available mining information discovered from each individual data block by a batch-mining algorithm. Examples of content attributes are number of transactions, number of mined patterns, and the set of previously mined large itemsets with supports.

The set of all patterns, with supports, previously mined from an individual data block is called a *pattern set* (ps) in this paper. Assume the minimum support is s and l large itemsets are discovered in a data block. A pattern set can be represented as $\mathrm{ps} = \{(x_i, s_i) \mid s_i \geqslant s \text{ and } 1 \leqslant i \leqslant l\}$, where x_i is a large itemset and s_i is its support. The pattern set is thus an essential content attribute of an inserted block of data.

A multidimensional pattern relation schema MPR with n_1 context attributes and n_2 content attributes can be represented as $\mathrm{MPR}(\mathrm{ID}, \mathrm{CX}_1, \mathrm{CX}_2, \ldots, \mathrm{CX}_{n_1}, \mathrm{CN}_1, \mathrm{CN}_2, \ldots, \mathrm{CN}_{n_2})$, where ID is an identification attribute, CX_i, $1 \leqslant i \leqslant n_1$, is a context attribute, and CN_i, $1 \leqslant i \leqslant n_2$, is a content attribute. Assume the multidimensional pattern relation **mpr** to be an instance of a given MPR that includes the tuples $\{t_1, t_2, \ldots, t_m\}$. Each tuple $t_i = (\mathrm{id}_i, \mathrm{cx}_{i1}, \mathrm{cx}_{i2}, \ldots, \mathrm{cx}_{in_1}, \mathrm{cn}_{i1}, \mathrm{cn}_{i2}, \ldots, \mathrm{cn}_{in_2})$ in **mpr** indicates that for the block of data identified by the contexts cx_{i1}, cx_{i2}, ..., and cx_{in_1}, the mined information contains cn_{i1}, cn_{i2}, ..., and cn_{in_2}.

Example 1. Table 1 shows a multidimensional pattern relation with the initial minimum support set at 5%. ID is the identification attribute, *Region*, *Branch*, and *Time* are context attributes, and *No_Trans*, *No_Patterns*, and *Pattern_Sets*

Table 1
A multidimensional pattern relation with minimum support = 5%

ID	Region	Branch	Time	No_Trans	No_Patterns	Pattern_Sets (itemset, support)
1	CA	San Francisco	2003/10	10,000	7	(A, 10%), (B, 11%), (C, 9%), (AB, 8%), (AC, 7%), (BC, 6%), (ABC, 6%)
2	CA	San Francisco	2003/11	15,000	3	(A, 5%), (B, 7%), (C, 5%)
3	CA	San Francisco	2003/12	12,000	2	(A, 5%), (C, 9%)
4	CA	Los Angeles	2003/10	20,000	4	(A, 8%), (B, 6%), (C, 7%), (AC, 6%)
5	CA	Los Angeles	2003/11	25,000	2	(A, 5%), (C, 6%)
6	CA	Los Angeles	2003/12	30,000	4	(A, 6%), (B, 6%), (C, 9%), (AB, 6%)
7	NY	New York	2003/10	18,000	3	(B, 8%), (C, 7%), (BC, 6%)
8	NY	New York	2003/11	18,500	2	(B, 8%), (C, 6%)
9	NY	New York	2003/12	19,000	5	(A, 5%), (B, 9%), (C, 8%), (D, 6%), (BC, 6%)

are content attributes. The *Pattern_Sets* attribute records the sets of large itemsets mined from previous data blocks. For example, the tuple ID = 1 shows that seven large itemsets, {(A, 10%), (B, 11%), (C, 9%), (AB, 8%), (AC, 7%), (BC, 6%), and (ABC, 6%)}, were discovered from 10,000 transactions and in the contexts of *Region* = **CA**, *Branch* = **San Francisco**, and *Time* = **2003/10**. The other tuples have similar meanings.

5. Multidimensional online mining for association rules

The goal of online mining is to find association rules satisfying the constraints in mining requests. The flexibility of mining requests can be increased by using the proposed multidimensional pattern relation. In this paper, an online mining approach called *three-phase online association rule mining* (TOARM) is proposed to carry out mining tasks with a multidimensional pattern relation. TOARM first selects tuples from the relation that satisfy the constraints in mining requests. It then integrates the mined information in these tuples and outputs them to users. Before describing the TOARM approach, we first formally define the problem to be solved and some related terminology. Some lemmas are also derived and proven.

Assume mpr = $\{t_1, t_2, \ldots, t_m\}$ is a multidimensional pattern relation based on an initial minimum support s. Given a mining request q with the set of contexts cx_q, the new minimum support s_q ($s_q \geqslant s$), and the new minimum confidence $conf_q$, the proposed algorithm will effectively and efficiently derive association

rules satisfying s_q, conf_q, and cx_q. A tuple with cx_q in a multidimensional pattern relation is called a *matched tuple*. Let t_i denote the ith tuple in a multidimensional pattern relation, t_i.trans the number of transactions in t_i, t_i.ps the pattern set in t_i, and $t_i.s_x$ the actual support of an itemset x in t_i. Lemma 1 is easily derived as follows.

Lemma 1. *For each itemset x satisfying s_q and cx_q in a mining request q, there exists at least a matched tuple t, such that $t.s_x$ satisfies s_q.*

Proof. We prove the lemma by contradiction. If $t_i.s_x < s_q$ for each matched tuple t_i, then

$$\sum_{t_i \in \text{matched tuples}} t_i.\text{trans} * t_i.s_x < \sum_{t_i \in \text{matched tuples}} t_i.\text{trans} * s_q.$$

It implies that the itemset x does not satisfy s_q, contradicting the claim that x satisfies s_q. Thus, there must exist at least a matched tuple t with $t.s_x \geqslant s_q$. □

According to Lemma 1, an itemset with support greater than or equal to s_q in at least one matched tuple is a possible candidate. The following lemma about *candidate itemsets* can thus be derived.

Lemma 2. *Each itemset x satisfying s_q and cx_q in a mining request q must be among the candidate itemsets obtained by collecting the ones whose supports are greater than or equal to s_q in at least one matched tuple.*

Example 2. For the multidimensional pattern relation given in Table 1, assume that a mining request q calls for getting the patterns under the contexts cx_q of *Region* = **CA** and *Time* = **2003/11–2003/12** and satisfying the minimum support s_q = 5.5%. The matched tuples are shown in Table 2. According to Lemma 2, the set of candidate itemsets is {{A}, {B}, {C}, {AB}}, which is the union of the itemsets appearing in the pattern sets with supports greater than 5.5%.

Table 2
Matched tuples in Example 2

ID	Region	Branch	Time	No_Trans	No_Patterns	Pattern_Sets (itemset, support)
2	CA	San Francisco	2003/11	15,000	3	(A, 5%), (B, 7%), (C, 5%)
3	CA	San Francisco	2003/12	12,000	2	(A, 5%), (C, 9%)
5	CA	Los Angeles	2003/11	25,000	2	(A, 5%), (C, 6%)
6	CA	Los Angeles	2003/12	30,000	4	(A, 6%), (B, 6%), (C, 9%), (AB, 6%)

The following relation can be derived for a candidate itemset x and its proper subsets.

Lemma 3. *If x is a candidate itemset, then $\forall x' \subset x$, x' is also a candidate itemset.*

Proof. If $x' \subset x$, then $t_i.s_{x'} \geqslant t_i.s_x$ for each tuple t_i in a multidimensional pattern relation. According to Lemma 2, if x is a candidate itemset, there must exist at least a matched tuple t with $t.s_x \geqslant s_q$. Thus, $t.s_{x'} \geqslant t.s_x \geqslant s_q$ for the tuple t. x' is thus a candidate itemset. □

The *appearing count* $\text{Count}_x^{\text{appearing}}$ of a candidate itemset x is defined as the count of x calculated from the matched tuples in which x appears. Thus

$$\text{Count}_x^{\text{appearing}} = \sum_{t_i \in \text{matched tuples and } x \in t_i.\text{ps}} t_i.\text{trans} * t_i.s_x. \tag{1}$$

The *upper-bound count* $\text{Count}_{\bar{x}}^{\text{UB}}$ of a candidate itemset x is defined as the upper bound count of x calculated from the matched tuples in which x does not appear. Thus

$$\text{Count}_{\bar{x}}^{\text{UB}} = \sum_{t_i \in \text{matched tuples and } x \notin t_i.\text{ps}} (t_i.\text{trans} * s - 1). \tag{2}$$

Let Match_Trans denote the number of transactions in the matched tuples. Thus

$$\text{Match\_Trans} = \sum_{t_i \in \text{matched tuples}} t_i.\text{trans}. \tag{3}$$

The *upper-bound support* s_x^{UB} of a candidate itemset x is thus calculated as

$$s_x^{\text{UB}} = \frac{\text{Count}_x^{\text{appearing}} + \text{Count}_{\bar{x}}^{\text{UB}}}{\text{Match\_Trans}}. \tag{4}$$

Lemma 4. *If x is a candidate itemset and s_x is its actual support, then $s_x \leqslant s_x^{\text{UB}}$.*

Proof

$$\begin{aligned} s_x &= \frac{\sum_{t_i \in \text{matched tuples}} t_i.\text{trans} * t_i.s_x}{\sum_{t_i \in \text{matched tuples}} t_i.\text{trans}} \\ &= \frac{\sum_{t_i \in \text{matched tuples and } x \in t_i.\text{ps}} t_i.\text{trans} * t_i.s_x + \sum_{t_i \in \text{matched tuples and } x \notin t_i.\text{ps}} t_i.\text{trans} * t_i.s_x}{\sum_{t_i \in \text{matched tuples}} t_i.\text{trans}} \end{aligned}$$

$$\leqslant \frac{\sum_{t_i \in \text{matched tuples and } x \in t_i.\text{ps}} t_i.\text{trans} * t_i.s_x + \sum_{t_i \in \text{matched tuples and } x \notin t_i.\text{ps}} (t_i.\text{trans} * s - 1)}{\sum_{t_i \in \text{matched tuples}} t_i.\text{trans}}$$

$$= \frac{\text{Count}_x^{\text{appearing}} + \text{Count}_{\bar{x}}^{\text{UB}}}{\text{Match\_Trans}} = s_x^{\text{UB}}.$$

Thus $s_x \leqslant s_x^{\text{UB}}$. □

Example 3. Continuing Example 2, the upper-bound supports of the four candidate itemsets {A}, {B}, {C}, and {AB}, are calculated as follows:

$$s_{\text{A}}^{\text{UB}} = \frac{\text{Count}_{\text{A}}^{\text{appearing}} + \text{Count}_{\overline{\text{A}}}^{\text{UB}}}{\text{Match\_Trans}} = \frac{15{,}000 * 5\% + 12{,}000 * 5\% + 25{,}000 * 5\% + 30{,}000 * 6\%}{15{,}000 + 12{,}000 + 25{,}000 + 30{,}000} = 0.0537,$$

$$s_{\text{B}}^{\text{UB}} = \frac{\text{Count}_{\text{B}}^{\text{appearing}} + \text{Count}_{\overline{\text{B}}}^{\text{UB}}}{\text{Match\_Trans}} = \frac{15{,}000 * 7\% + 30{,}000 * 6\% + 12{,}000 * 5\% - 1 + 25{,}000 * 5\% - 1}{15{,}000 + 12{,}000 + 25{,}000 + 30{,}000} = 0.0573,$$

$$s_{\text{C}}^{\text{UB}} = \frac{\text{Count}_{\text{C}}^{\text{appearing}} + \text{Count}_{\overline{\text{C}}}^{\text{UB}}}{\text{Match\_Trans}} = \frac{15{,}000 * 5\% + 12{,}000 * 9\% + 25{,}000 * 6\% + 30{,}000 * 9\%}{15{,}000 + 12{,}000 + 25{,}000 + 30{,}000} = 0.0735,$$

and

$$s_{\text{AB}}^{\text{UB}} = \frac{\text{Count}_{\text{AB}}^{\text{appearing}} + \text{Count}_{\overline{\text{AB}}}^{\text{UB}}}{\text{Match\_Trans}} = \frac{30{,}000 * 6\% + 15{,}000 * 5\% - 1 + 12{,}000 * 5\% - 1 + 25{,}000 * 5\% - 1}{15{,}000 + 12{,}000 + 25{,}000 + 30{,}000} = 0.0536.$$

Lemma 5. *If x is a candidate itemset, then $\forall x' \subset x$, $s_{x'}^{\text{UB}} \geqslant s_x^{\text{UB}}$.*

Proof. If $x' \subset x$, then $t_i.s_{x'} \geqslant t_i.s_x$ for each tuple t_i in a multidimensional pattern relation. Therefore

$$
\begin{aligned}
s_{x'}^{\mathrm{UB}} &= \left(\mathrm{Count}_{x'}^{\mathrm{appearing}} + \mathrm{Count}_{\bar{x}'}^{\mathrm{UB}}\right) \Big/ \mathrm{Match\_Trans} \\
&= \left(\sum_{t_i \in \text{matched tuples and } x' \in t_i.\mathrm{ps}} t_i.\mathrm{trans} * t_i.s_{x'} \right. \\
&\quad \left. + \sum_{t_i \in \text{matched tuples and } x' \notin t_i.\mathrm{ps}} (t_i.\mathrm{trans} * s - 1) \right) \Big/ \left(\sum_{t_i \in \text{matched tuples}} t_i.\mathrm{trans} \right) \\
&= \left(\sum_{t_i \in \text{matched tuples and } x' \in t_i.\mathrm{ps} \text{ and } x \in t_i.\mathrm{ps}} t_i.\mathrm{trans} * t_i.s_{x'} \right. \\
&\quad + \sum_{t_i \in \text{matched tuples and } x' \in t_i.\mathrm{ps} \text{ and } x \notin t_i.\mathrm{ps}} t_i.\mathrm{trans} * t_i.s_{x'} \\
&\quad \left. + \sum_{t_i \in \text{matched tuples and } x' \notin t_i.\mathrm{ps}} (t_i.\mathrm{trans} * s - 1) \right) \Big/ \left(\sum_{t_i \in \text{matched tuples}} t_i.\mathrm{trans} \right) \\
&\geqslant \left(\sum_{t_i \in \text{matched tuples and } x' \in t_i.\mathrm{ps} \text{ and } x \in t_i.\mathrm{ps}} t_i.\mathrm{trans} * t_i.s_{x} \right. \\
&\quad + \sum_{t_i \in \text{matched tuples and } x' \in t_i.\mathrm{ps} \text{ and } x \notin t_i.\mathrm{ps}} (t_i.\mathrm{trans} * s - 1) \\
&\quad \left. + \sum_{t_i \in \text{matched tuples and } x' \notin t_i.\mathrm{ps}} (t_i.\mathrm{trans} * s - 1) \right) \Big/ \left(\sum_{t_i \in \text{matched tuples}} t_i.\mathrm{trans} \right) \\
&= \left(\sum_{t_i \in \text{matched tuples and } x \in t_i.\mathrm{ps}} t_i.\mathrm{trans} * t_i.s_{x} \right. \\
&\quad \left. + \sum_{t_i \in \text{matched tuples and } x \notin t_i.\mathrm{ps}} (t_i.\mathrm{trans} * s - 1) \right) \Big/ \left(\sum_{t_i \in \text{matched tuples}} t_i.\mathrm{trans} \right) \\
&= \left(\mathrm{Count}_{x}^{\mathrm{appearing}} + \mathrm{Count}_{\bar{x}}^{\mathrm{UB}}\right) / (\mathrm{Match\_Trans}) = s_x^{\mathrm{UB}}.
\end{aligned}
$$

Thus, $s_{x'}^{\mathrm{UB}} \geqslant s_x^{\mathrm{UB}}$. □

Lemma 6. *If a candidate itemset x is contained in all matched tuples, then* $s_x^{\mathrm{UB}} = s_x$.

Proof. If x is contained in all the matched tuples, then

$$
s_x^{\mathrm{UB}} = \frac{\mathrm{Count}_x^{\mathrm{appearing}} + \mathrm{Count}_{\bar{x}}^{\mathrm{UB}}}{\mathrm{Match\_Trans}} = \frac{\sum_{t_i \in \text{matched tuples}} t_i.\mathrm{trans} * t_i.s_x}{\sum_{t_i \in \text{matched tuples}} t_i.\mathrm{trans}} = s_x. \qquad \square
$$

Example 4. Continuing Examples 2 and 3, according to Lemmas 4 and 5, candidate itemsets {A} and {AB} will be pruned since {AB} is a proper superset of {A} and the upper-bound support of {A} is less than s_q (=5.5%). According to Lemma 6, the candidate itemset {C} will be put into the set of final large itemsets since it appears in all matched tuples and its support is greater than 5.5%. Only the remaining candidate itemset {B} needs further processing.

6. Three-phase online association rule mining (TOARM)

In this section, we propose the TOARM approach for carrying out mining tasks with a multidimensional pattern relation. It consists of three main phases, *candidate itemset generation*, *candidate itemset reduction*, and *association rule generation*. The *candidate itemset generation* phase selects tuples that satisfy the context constraints in mining requests and generates candidate itemsets from matched tuples. The *candidate itemset reduction* phase then calculates the upper-bound supports for the candidate itemsets and uses two pruning strategies to reduce the number of candidates. Finally, the *association rule generation* phase finds final large itemsets and derives association rules from them. The proposed three-phase online mining approach is described below.

The three-phase online association rule mining (TOARM) approach

Input: A multidimensional pattern relation based on an initial minimum support s and a mining request q with a context set cx_q, a minimum support s_q and a minimum confidence conf_q.

Output: A set of association rules satisfying the mining request q.

Phase 1 Candidate itemset generation:

(a) Select tuples satisfying cx_q from the multidimensional pattern relation.

(b) Gather the candidate itemsets appearing in the matched tuples.

(c) Calculate $\mathrm{Count}_x^{\mathrm{appearing}}$ and $\mathrm{Count}_{\bar{x}}^{\mathrm{UB}}$ for each candidate itemset x.

Phase 2 Candidate itemset reduction:

(a) Calculate the upper-bound support s_x^{UB} for each candidate itemset x using:

$$s_x^{\mathrm{UB}} = \frac{\mathrm{Count}_x^{\mathrm{appearing}} + \mathrm{Count}_{\bar{x}}^{\mathrm{UB}}}{\mathrm{Match\_Trans}}.$$

(b) Discard candidate itemset x and its proper supersets if $s_x^{UB} < s_q$.

(c) Put x into the set of large itemsets if $s_x^{UB} = \frac{Count_x^{appearing}}{Match\_Trans}$ and $s_x^{UB} \geqslant s_q$.

Phase 3 Association rule generation:

(a) Check whether each remaining candidate itemset x is large by scanning the underlying blocks of data for the matched tuples in which x does not appear.

(b) Generate association rules satisfying the minimum confidence $conf_q$ from the set of large itemsets.

The TOARM approach considers only itemsets appearing in matched tuples and satisfying minimum support as candidates. It also uses two pruning strategies to reduce the number of candidate itemsets. It therefore only needs to reprocess the remaining candidate itemsets against the underlying blocks of data for matched tuples in which they do not appear. For this reason, the cost of reprocessing underlying blocks of data by the TOARM approach is less than that of typical batch mining or incremental mining approaches (experimental results presented below show this).

Theorem 1. *The TOARM approach can correctly obtain association rules in response to an online mining request q as long as its minimum support s_q is greater than or equal to the initial minimum support s for getting the multidimensional pattern relation.*

Proof. According to Lemma 2, all candidate itemsets for q are collected in Phase 1 of the TOARM approach. After that, the candidate itemsets whose upper-bound supports are less than s_q are pruned in Phase 2(b) of the TOARM approach according to Lemmas 4 and 5. Also, the candidate itemsets which appear in all the matched tuples can know their actual supports according to Lemma 6. If they satisfy s_q, they are put into the set of final large itemsets in Phase 2(c) of the TOARM approach. Finally, the actual supports of the remaining candidate itemsets can be found by Phase 3(a) of the TOARM approach from the underlying blocks of data. The final large itemsets can then be determined. The association rules can thus be derived by Phase 3(b) of the TOARM approach. □

7. Implementation consideration

7.1. The data structure used for the TOARM approach

The TOARM approach utilizes a *latticed* data structure [1,21] to maintain candidate itemsets. The lattice is constructed as follows. For each candidate

itemset x, a corresponding vertex c_x is built in the lattice C. Vertex c_x consists of two integers, $\text{Count}_x^{\text{appearing}}$ and $\text{Count}_{\bar{x}}^{\text{UB}}$, that record, respectively, its appearing and upper-bound counts. For any pair of vertices c_x and c_y corresponding to candidate itemsets x and y, there is a directed edge from c_x to c_y if x is a *parent* of y. An itemset x is said to be a parent of an itemset y if y can be obtained by adding an item to x, and inversely, y is said to be a child of x. Therefore, a candidate itemset may have more than one parent and more than one child in the constructed lattice.

Example 5. Consider the candidate itemsets illustrated in Example 2. The lattice constructed according to the candidate itemsets is illustrated in Fig. 2, where the vertex labeled "Null" denotes the greatest lower bound of the lattice.

7.2. Phase 1: Candidate itemset generation

Two scans are needed in this phase to construct the lattice and calculate the appearing and upper-bound counts of the candidate itemsets. The first scan checks the multidimensional pattern relation and the second scan checks only the matched tuples. During the first scan, tuples satisfying the context constraints given in the mining request are selected, and all the itemsets in the matched tuples with supports greater than or equal to the minimum support are collected and used to construct a corresponding lattice. During the second scan, the $\text{Count}_x^{\text{appearing}}$ and $\text{Count}_{\bar{x}}^{\text{UB}}$ values of each candidate itemset x are accumulated as each matched tuple is processed. After all matched tuples have been processed, the phase then generates all candidate itemsets with appearing

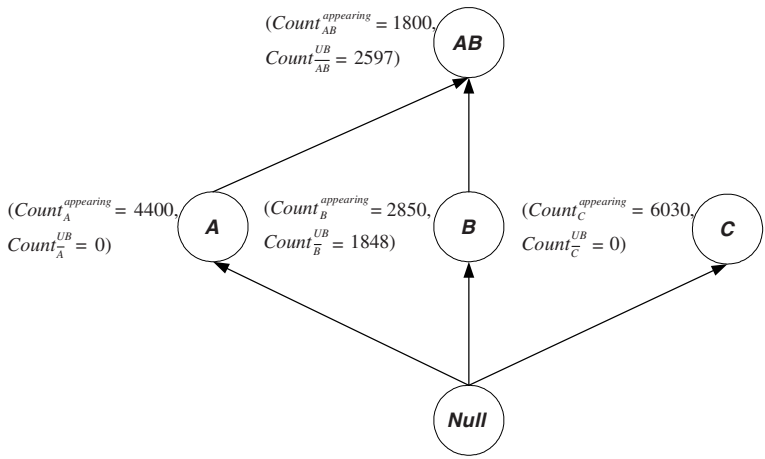

Fig. 2. Lattice constructed according to the candidate itemsets in Example 2.

and upper-bound counts corresponding to the mining request. The algorithm for this proposed phase is described below.

Phase 1: The candidate-itemset-generation algorithm

Input: A multidimensional pattern relation based on an initial minimum support s, and a mining request q with a set of contexts cx_q and a minimum support s_q.

Output: A lattice C recording information on candidate itemsets.

Step 1: Set $C = \emptyset$ and Match_Trans = 0, where C is used to maintain the set of candidate itemsets and Match_Trans is used to store total number of transactions in the matched tuples processed.

Step 2: For each tuple t in the given multidimensional pattern relation, do the following substeps:

Step 2-1: If t satisfies cx_q, put it in the matched set and do Step 2-2.

Step 2-2: For each itemset $x \in t.\mathrm{ps}$, if $x \notin C$ and $t.s_x \geqslant s_q$, put x into C, set $\mathrm{Count}_x^{\mathrm{appearing}} = 0$, $\mathrm{Count}_x^{\mathrm{UB}} = 0$, and add appropriate edges to its parents and children.

Step 3: For each tuple t in the matched set, do the following substeps:

Step 3-1: Set ProcessedSet = $\emptyset$, where ProcessedSet is a set used to store the processed itemsets in C.

Step 3-2: For each itemset $x \in t.\mathrm{ps}$, if $x \in C$, set $\mathrm{Count}_x^{\mathrm{appearing}} = \mathrm{Count}_x^{\mathrm{appearing}} + t.\mathrm{trans} * t.s_x$ and ProcessedSet = ProcessedSet $\cup \{x\}$.

Step 3-3: If $|\mathrm{ProcessedSet}| \neq |C|$, for each remaining itemset $x \in C$, set $\mathrm{Count}_x^{\mathrm{UB}} = \mathrm{Count}_x^{\mathrm{UB}} + t.\mathrm{trans} * s - 1$.

Step 3-4: Set Match_Trans = Match_Trans + t.trans.

Step 4: Output C to Phase 2.

After Step 4, a lattice recording the candidate itemsets, and appearing and upper-bound counts from the given multidimensional pattern relation is generated.

Example 6. Consider the matched tuples in Table 2 and the mining request in Example 2. After the first scan, the *candidate-itemset-generation* algorithm constructs a lattice, as shown in Fig. 3. When processing the first tuple in Table 2 during the second scan, the algorithm updates the information on the candidate itemsets {A}, {B} and {C} in the lattice. The updated lattice after first-tuple processing is shown in Fig. 4.

7.3. Phase 2: Candidate itemset reduction

In this phase, the proposed level-wise *candidate-itemset-reduction* algorithm uses two pruning strategies to reduce the number of candidate itemsets in the

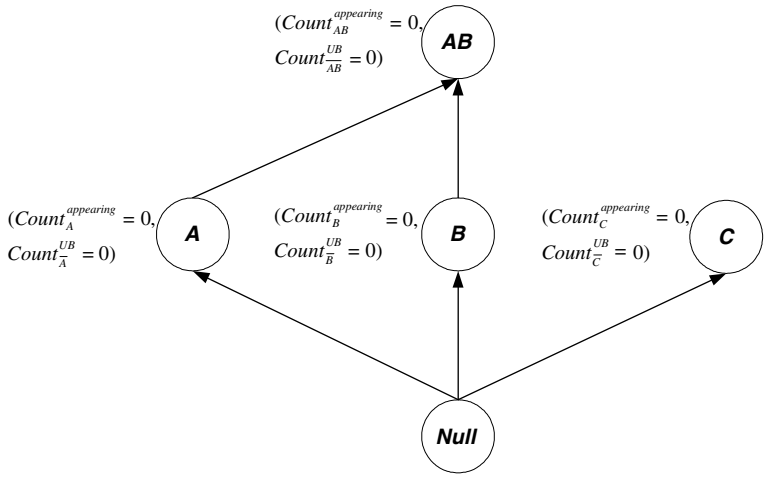

Fig. 3. Lattice after the first scan.

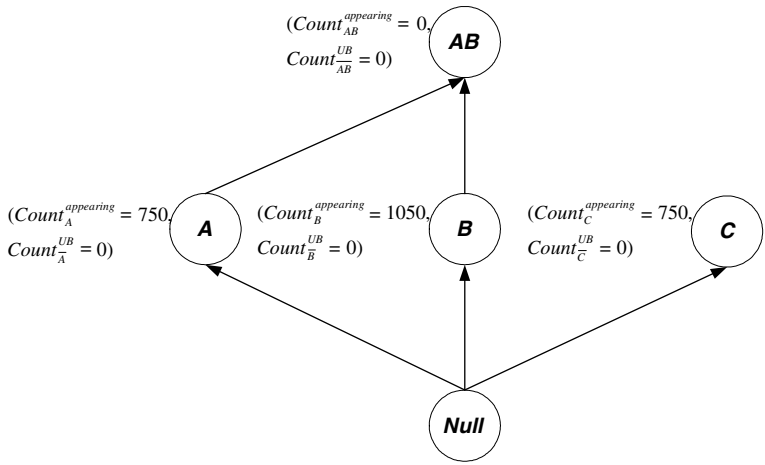

Fig. 4. Lattice after processing the first tuple in Table 2 during the second scan.

lattice. The proposed algorithm first checks candidate 1-itemsets that contain only one item. If the upper-bound support of a 1-itemset is less than the minimum support in the mining request, the first pruning strategy removes it and all its proper supersets from the lattice. If a 1-itemset appears in all matched tuples, and its upper-bound support is greater than or equal to the minimum support in the mining request, the second pruning strategy puts it in the set

of final large itemsets and also removes it from the lattice. This procedure is then repeated for all itemsets containing two or more items until all candidate itemsets in the lattice have been processed. The *candidate-itemset-reduction* algorithm then produces a smaller set of candidate itemsets for further processing in Phase 3. The proposed algorithm for this is described below.

Phase 2: The candidate-itemset-reduction algorithm

Input: The lattice C from Phase 1 and a minimum support s_q.

Output: A pruned lattice C.

Step 1: Set $k = 1$, where k is used to store the number of items in the candidate itemset currently being processed.

Step 2: Do the following substeps for each itemset $x \in C_k$:

Step 2-1: Calculate the upper-bound support s_x^{UB} using the formula:

$$s_x^{\mathrm{UB}} = \frac{\mathrm{Count}_x^{\mathrm{appearing}} + \mathrm{Count}_{\bar{x}}^{\mathrm{UB}}}{\mathrm{Match\_Trans}}.$$

Step 2-2: If $s_x^{\mathrm{UB}} < s_q$, set $C = C - \{y | y \in C \text{ and } x \subseteq y\}$.

Step 2-3: If $s_x^{\mathrm{UB}} = \frac{\mathrm{Count}_x^{\mathrm{appearing}}}{\mathrm{Match\_Trans}}$ and $s_x^{\mathrm{UB}} \geqslant s_q$, then set $L = L \cup \{x\}$ and $C = C - \{x\}$.

Step 3: Set $k = k + 1$.

Step 4: Repeat Steps 2 to 3 until all candidate itemsets have been processed.

Step 5: Return C.

7.4. Phase 3: Association rule generation

After Phase 2, all remaining candidate itemsets in the lattice have sufficient upper-bound support values but do not appear in at least one matched tuple. The proposed *association-rule-generation* algorithm thus must re-process the underlying blocks of data for these tuples to get their actual supports. The support of a remaining itemset x in the lattice can easily be calculated using the following formula:

$$s_x \frac{\mathrm{Count}_x^{\mathrm{appearing}} + \mathrm{Count}_{\bar{x}}^{\mathrm{appearing}}}{\mathrm{Match\_Trans}},$$

where $\mathrm{Count}_{\bar{x}}^{\mathrm{appearing}} = \sum_{t_i \in \text{matched tuples and } x \notin t_i.\mathrm{ps}} t_i.\mathrm{count}_x$ and $t_i.\mathrm{count}_x$ is the actual x count obtained by re-processing the block of data indicated by t_i. The

association-rule-generation algorithm also processes the remaining candidate itemsets in a level-wise way. It first processes 1-itemsets in the lattice. If the actual support of a candidate 1-itemset is less than the minimum support in the mining request, it and its proper supersets are removed from the lattice as in Phase 2. Otherwise, the 1-itemset is a large itemset for the mining request. This procedure is then repeated for itemsets with more items until all the remaining itemsets in the lattice have been processed. After the final large itemsets have been found, association rules can then easily be generated from them. The proposed algorithm for this is described below.

Phase 3: The association-rule-generation algorithm

Input: The lattice C from Phase 2, a minimum support s_q, and a minimum confidence conf_q.

Output: A set of association rules satisfying the mining request q.

Step 1: Set $k = 1$, where k is used to store the number of items in the candidate itemset currently being processed.

Step 2: For the underlying block of data D_i indicated by each matched tuple t_i, if there is at least one remaining candidate itemset not appearing in t_i, do the following substeps.

Step 2-1: For each remaining candidate itemset $x \in C_k$ where x does not appear in t_i, count $t_i.\text{count}_x$ by loading and rescanning D_i.

Step 2-2: Set $\text{Count}_{\bar{x}}^{\text{appearing}} = \text{Count}_{\bar{x}}^{\text{appearing}} + t_i.\text{count}_x$.

Step 3: For each $x \in C_k$, calculate the actual support of x using this formula:

$$s_x = \frac{\text{Count}_x^{\text{appearing}} + \text{Count}_{\bar{x}}^{\text{appearing}}}{\text{Match\_Trans}}.$$

Step 4: If $s_x < s_q$, then set $C = C - \{y \mid y \in C \text{ and } x \subseteq y\}$. Otherwise, set $L = L \cup \{x\}$ and $C = C - \{x\}$.

Step 5: Set $k = k + 1$.

Step 6: Repeat Steps 2 to 5 until all the candidate itemsets have been processed.

Step 7: Derive the association rules satisfying conf_q from the set of large itemsets L.

8. Experiments

Before presenting the experimental results, we first describe the experimental environments and the datasets used.

8.1. Experimental environment and datasets used

Our experiments were conducted in Java on a workstation with dual XEON 2.8 GHz processors and 2048 MB main memory, running the RedHat 9.0 operating system. Several synthetic datasets and a real-world dataset called *BMS-POS* [40] were used. The synthetic datasets were generated by a generator similar to that used in [4], considering the parameters listed in Table 3. The generator first generated L maximal potentially large itemsets, each with an average of I items. The items in the potentially large itemsets were randomly chosen from the total N items according to their actual sizes. The generator then generated D transactions, each with an average of T items. The items in a transaction were generated according to the L maximal potentially large itemsets in a probabilistic way. Details of the dataset generation process may be found in [4].

The four groups of synthetic datasets generated and used in our experiments are listed in Table 4, where datasets in the same group had the same D, T and I values, but different L or N values. Each dataset was treated as a block of data in the database. For example, Group 1 in Table 4 contained ten blocks of data, from $T10I8D10KL^1$ to $T10I8D10KL^{10}$, each consisting of 10,000 transactions averaging 10 items and generated according to 200–245 maximal potentially large itemsets with an average size of 8 from a total of 100 items. Let a heterogeneous dataset be defined as one in which the data subsets forming the tuples in a multidimensional pattern relation have different items. Among the four groups, Group 2 may be considered heterogeneous because its varied N values

Table 3
Parameters considered when generating datasets

Parameter	Description
D	Number of transactions
N	Number of items
L	Number of maximal potentially large itemsets
T	Average size of items in transactions
I	Average size of items in maximal potentially large itemsets

Table 4
The four groups of synthetic datasets

Group	Size	Datasets	D	T	I	L	N
1	10	$T10I8D10KL^1$ to $T10I8D10KL^{10}$	10,000	10	8	200 to 245	100
2	10	$T10I8D10KN^1$ to $T10I8D10KN^{10}$	10,000	10	8	200	100 to 145
3	10	$T20I8D100KL^1$ to $T20I8D100KL^{10}$	100,000	20	8	400 to 490	200
4	5	$T10I8D500KL^1$ to $T10I8D500KL^5$	500,000	10	8	400 to 560	200

yield different items. This group of datasets was used to show how our approach dealt with heterogeneous blocks of data.

The *BMS-POS* dataset contains several years of point-of-sale data from a large electronics retailer. Each transaction in this dataset is a customer purchase transaction consisting of all the product categories purchased at one time. There are 515,597 transactions in the dataset. The number of distinct items is 1657, the maximal transaction size is 164, and the average transaction size is 6.5. This dataset was also used in the KDDCUP 2000 competition. In our experiments, the fifth group of data consisted of ten equal-size data subsets partitioned from the *BMS-POS* dataset.

8.2. Experimental results

Multidimensional pattern relations were first derived from each group of datasets. These are summarized in Table 5.

Two batch-based mining algorithms, Apriori and Partition, and one incremental mining algorithm, FUP, in addition to our proposed TOARM algorithm, were run on Groups 1 to 5 along with various minimum supports in the mining requests. The Partition algorithm partitioned the data sets according to group size (the number of datasets in a group). The FUP algorithm treated each dataset in a group as a new addition of transactions. Execution times for the four algorithms on the synthetic data in Groups 1 to 4 are shown in Fig. 5.

We first compare the TOARM algorithm with the Apriori algorithm. Fig. 5(a), (c), and (d) shows that execution times for the TOARM algorithm on Groups 1, 3, and 4 were always much less than those of the Apriori algorithm. This is because the datasets in these three groups were homogeneous, meaning they used the same set of items in each group. In this situation, the number of candidate itemsets considered by the TOARM algorithm was much closer to the number of final large itemsets than those considered by the Apriori

Table 5
Mining information for the five groups

Group	Initial minimum support (%)	Average length of maximal large itemsets	Average size of large itemsets
1	2	11	9006
2	2	9	5093
3	2	9	12,127
4	2	5	799
5	1	5	1300

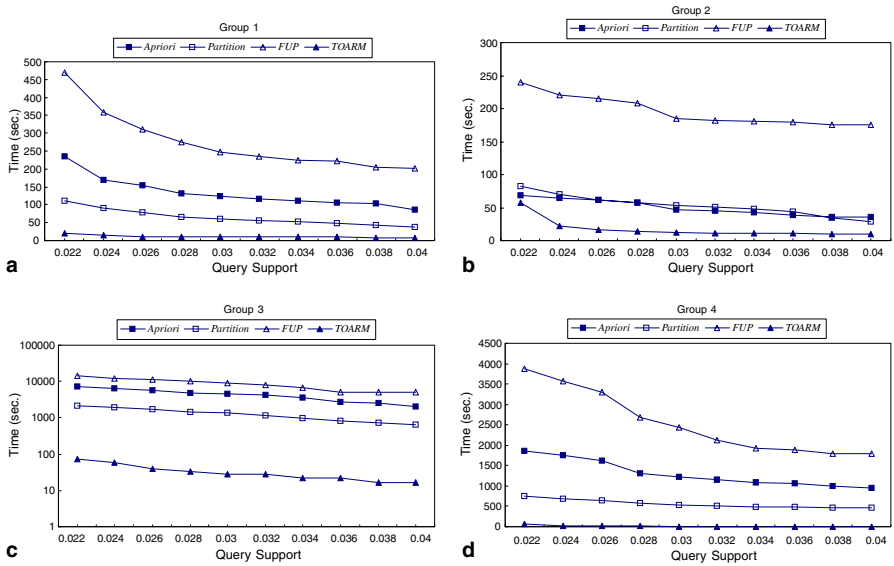

Fig. 5. Execution times for the four algorithms on Groups 1 to 4.

algorithm. The former thus had more compact candidate sets than the latter. For example, Table 6 shows the number of candidate itemsets considered by the TOARM and the Apriori algorithms for Group 4 with minimum supports ranging from 0.022 to 0.04 in the mining requests.

By contrast, the datasets in Group 2 were heterogeneous, meaning they used different sets of items. In this situation, the number of candidate itemsets considered by the TOARM algorithm was much larger than the number of final large itemsets considered by the Apriori algorithm since most of the candidate itemsets appeared in only one or a few tuples in the multidimensional pattern relation. Table 7 shows the number of candidate itemsets considered by the TOARM and Apriori algorithms for Group 2 along with minimum supports

Table 6
The numbers of candidate itemsets for Group 4

Approach	Support									
	0.022	0.024	0.026	0.028	0.03	0.032	0.034	0.036	0.038	0.04
Number of candidate itemsets										
TOARM	959	690	550	442	372	308	269	241	220	199
Apriori	11,636	10,327	9165	8590	7722	7085	6603	6346	5898	5369
Number of final large itemsets										
TOARM/Apriori	574	453	373	318	260	228	201	177	158	144

Table 7
The numbers of candidate itemsets for Group 2

Approach	Support									
	0.022	0.024	0.026	0.028	0.03	0.032	0.034	0.036	0.038	0.04
Number of candidate itemsets										
TOARM	20,893	16,003	11,920	9016	7421	6541	5731	4984	3775	2816
Apriori	11,615	10,157	9158	8016	7372	6704	6070	5243	4593	4255
Number of final large itemsets										
TOARM/Apriori	902	778	684	608	537	473	417	372	327	296

ranging from 0.022 to 0.04 in the mining requests. Table 7 also shows that while the number of candidate itemsets for Group 2 considered by the TOARM algorithm was larger than that considered by the Apriori algorithm, the TOARM algorithm used two pruning strategies in Phase 2 and thus only had to re-process the remaining candidate itemsets against the underlying datasets in Phase 3. The result was that the TOARM algorithm usually required less time than the Apriori algorithm. This is consistent with the results shown in Fig. 5(b).

Next, we compare the TOARM algorithm with the Partition algorithm. Although the number of candidate itemsets considered by the Partition algorithm in the second pass was equal to that considered by the TOARM algorithm, the Partition algorithm must generate a set of all potentially large itemsets from each partition during its first pass. The TOARM algorithm can, however, use the pattern sets in the multidimensional pattern relation to achieve this purpose. Therefore, the execution times required by the TOARM algorithm on Groups 1 to 4 were always less than those required by the Partition algorithm. This is also consistent with the results shown in Fig. 5.

Finally, we compare the TOARM algorithm with the FUP algorithm. The FUP algorithm can, in general, perform well when the size of newly inserted transactions is relatively smaller than the size of an original database because the cost of generating candidate itemsets from only new transactions is usually low and a large proportion of the candidate itemsets can be determined from previously mined large itemsets. However, the FUP algorithm treated the datasets in each of our application groups as increments and yielded even worse performance than the Apriori algorithm, especially on the heterogeneous datasets since it had to process all of them one by one. Fig. 5(a), (c), and (d) shows that the execution times for the FUP algorithm on the three homogeneous groups were about twice those of the Apriori algorithm. On the second group, which was heterogeneous, the FUP algorithm required about four times the execution time required by the Apriori algorithm.

The execution times for the four algorithms on the real-world *BMS-POS* [40] dataset used in the second part of the experiments are shown in Fig. 6. The

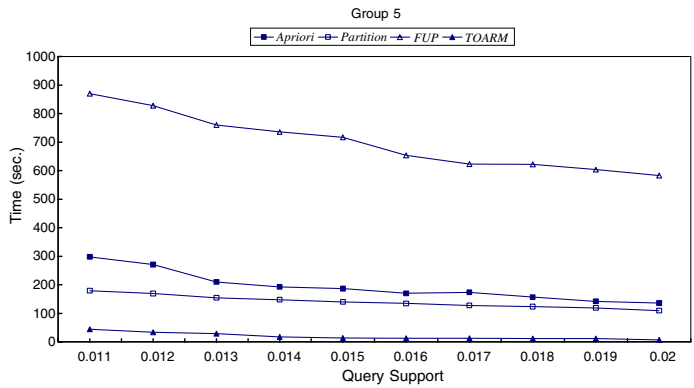

Fig. 6. Execution times for the four algorithms on the real-world *BMS-POS* dataset.

TOARM algorithm had the best performance among the four approaches. Experiments were then made to show the influence of the initial minimum support used in constructing the multidimensional pattern relation on performance. Execution times for the TOARM algorithm on Groups 1 and 2 using various initial minimum supports with the minimum support in mining requests fixed at 0.023 are shown in Fig. 7. Note that the execution times by the TOARM algorithm increased nonlinearly along with the initial minimum support because higher initial minimum supports meant a larger number of remaining candidate itemsets in Phase 3. The numbers of remaining candidate itemsets after Phase 2 along with various initial minimum supports for Groups 1 and 2 are shown in Fig. 8. The results are quite consistent with the discussion above.

Finally, the scalability aspect was examined by running the TOARM and Apriori algorithms on Group 1 using various numbers of blocks ranging from

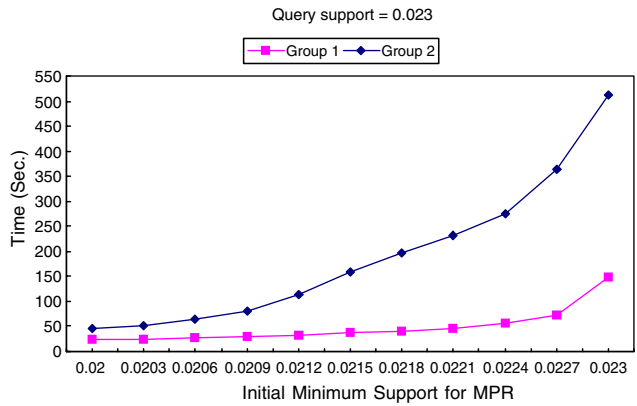

Fig. 7. Execution time vs. initial minimum supports for Groups 1 and 2.

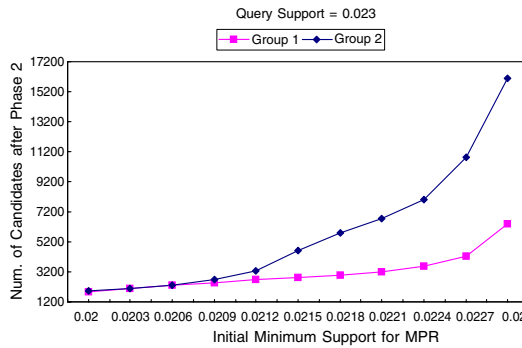

Fig. 8. Numbers of candidate itemsets after Phase 2 vs. initial minimum supports for Groups 1 and 2.

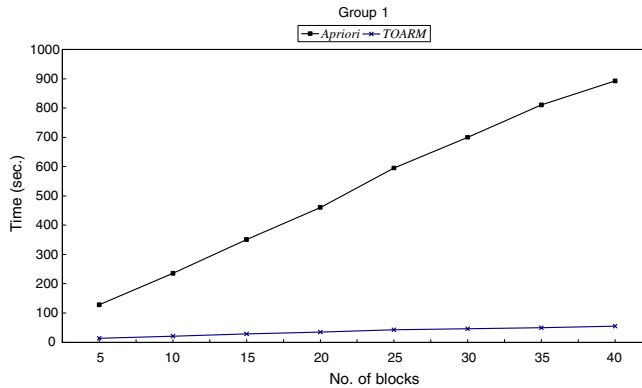

Fig. 9. Execution times on various numbers of blocks for the TOARM and Apriori algorithms.

5 to 40, with the minimum support in mining requests set at 0.022. The execution times required by the two algorithms are shown in Fig. 9. It is clear that the execution times required by the TOARM algorithm for various numbers of blocks were small, and seemed to grow slowly and linearly with the numbers of blocks.

9. Conclusion and future work

In this paper, we have extended the concept of effectively utilizing patterns previously discovered for online decision support under multidimensional considerations. By structurally and systematically storing additional context and mining information in the multidimensional pattern relation, our proposed

TOARM approach easily and efficiently derives association rules that satisfy diverse user-concerned constraints. Our experimental results show the proposed TOARM approach is more efficient than the well known Apriori, Partition, and FUP approaches under the considerations mentioned above.

Our approach requires storage of multidimensional pattern relations, which leads to some overhead in creating and maintaining this structure. The most time-consuming process is the generation of the pattern sets for these relations. However, the proposed approach is designed for applications in which data is inserted and analyzed in blocks during specific time intervals and pattern sets may have to be generated during each time interval for analysis. Such applications are frequently found in both business and industry.

For heterogeneous datasets, the TOARM approach may generate more candidate itemsets than a level-wise candidate generation algorithm (such as the Apriori algorithm) because most candidate itemsets appear in only one or few matched tuples. If necessary, the TOARM approach may be easily modified to operate in a level-wise way to deal with this problem at the expense of I/O cost. Since the TOARM approach is based on pre-defined multidimensional pattern relations, its power is limited by the contents stored in the relations. If mining requests have smaller minimum supports than the initial minimum support or ask for additional contexts apart from those in the multidimensional pattern relation, the TOARM approach may need to be modified to work well. In the former case, additional data scans may be required to get correct pattern sets; in the latter case, more context information may need to be stored in multidimensional pattern relations or appropriate database join operations may be required to include context information from other tables. These issues are interesting and may be studied in the future. We will also attempt to use other techniques to further improve the performance of the proposed methodology. For example, we can construct an Iceberg cube [8,15] or use materialized views [11,37] for the proposed multidimensional pattern relation to provide more efficient online association rule generation and more powerful mining services.

Acknowledgement

This research was supported by the National Science Council of the Republic of China under Grant No. NSC93-2752-E-009-006-PAE.

References

[1] C.C. Aggarwal, P.S. Yu, A new approach to online generation of association rules, IEEE Transactions on Knowledge and Data Engineering 13 (4) (2001) 527–540.

[2] R. Agrawal, T. Imielinksi, A. Swami, Mining association rules between sets of items in large database, in: ACM SIGMOD Conference, Washington, DC, USA, 1993, pp. 207–216.
[3] R. Agrawal, T. Imielinksi, A. Swami, Database mining: a performance perspective, IEEE Transactions on Knowledge and Data Engineering 5 (6) (1993) 914–925.
[4] R. Agrawal, R. Srikant, Fast algorithm for mining association rules, in: ACM International Conference on Very Large Data Bases, 1994, pp. 487–499.
[5] R. Agrawal, R. Srikant, Mining sequential patterns, in: IEEE International Conference on Data Engineering, 1995, pp. 3–14.
[6] W.G. Aref, M.G. Elfeky, A.K. Elmagarmid, Incremental, online, and merge mining of partial periodic patterns in time-series databases, IEEE Transactions on Knowledge and Data Engineering 16 (3) (2004) 332–342.
[7] R.J. Bayardo, R. Agrawal, D. Gunopulos, Constraint-based rule mining in large, dense databases, in: IEEE International Conference on Data Engineering, 1999, pp. 188–197.
[8] K. Beyer, R. Ramakrishnan, Bottom-up computation of sparse and iceberg cubes, in: ACM SIGMOD Conference, 1999, pp. 359–370.
[9] S. Brin, R. Motwani, C Silverstein, Beyond market baskets: generalizing association rules to correlations, in: ACM SIGMOD Conference, Tucson, AZ, USA, 1997, pp. 265–276.
[10] S. Brin, R. Motwani, J.D. Ullman, S. Tsur, Dynamic itemset counting and implication rules for market basket data, in: ACM SIGMOD Conference, Tucson, AZ, USA, 1997, pp. 255–264.
[11] S. Chaudhuri, U. Dayal, An overview of data warehousing and OLAP technology, ACM SIGMOD Record 26 (1997) 65–74.
[12] M.S. Chen, J. Han, P.S. Yu, Data mining: an overview from database perspective, IEEE Transactions on Knowledge and Data Engineering 8 (6) (1996) 866–883.
[13] D.W. Cheung, J. Han, V.T. Ng, C.Y. Wong, Maintenance of discovered association rules in large databases: an incremental updating approach, in: IEEE International Conference on Data Engineering, 1996, pp. 106–114.
[14] D.W. Cheung, S.D. Lee, B. Kao, A general incremental technique for maintaining discovered association rules, in: Proceedings of Database Systems for Advanced Applications, Melbourne, Australia, 1997, pp. 185–194.
[15] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, J.D. Ullman, Computing iceberg queries efficiently, in: ACM International Conference on Very Large Data Bases, 1998, pp. 299–310.
[16] R. Feldman, Y. Aumann, A. Amir, H. Mannila, Efficient algorithms for discovering frequent sets in incremental databases, ACM SIGMOD Workshop on DMKD, USA, 1997, pp. 59–66.
[17] G. Grahne, L.V.S. Lakshmanan, X. Wang, M.H. Xie, On dual mining: from patterns to circumstances and back, in: IEEE International Conference on Data Engineering, 2001, pp. 195–204.
[18] J. Han, L.V.S. Lakshmanan, R. Ng, Constraint-based, multidimensional data mining, IEEE Computer Magazine (1999) 2–6.
[19] J. Han, M. Kamber, Data mining: concepts and techniques, Morgan Kaufmann Publishers, Los Altos, CA, 2001.
[20] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: ACM SIGMOD Conference, 2000, pp. 1–12.
[21] C. Hidber, Online association rule mining, in: ACM SIGMOD Conference, USA, 1999, pp. 145–156.
[22] T.P. Hong, C.Y. Wang, Y.H. Tao, A new incremental data mining algorithm using pre-large itemsets, International Journal on Intelligent Data Analysis (2001).
[23] W.H. Immon, Building the data warehouse, Wiley Computer Publishing, 1996.

[24] M. Kamber, J. Han, J.Y. Chiang, Metarule-guided mining of multi-dimensional association rules using data cubes, in: The International Conference on Knowledge Discovery and Data Mining, 1997, pp. 207–210.
[25] H. Kona, S. Chakravarthy, Partitioned approach to association rule mining over multiple databases, in: The International Conference on Data Warehousing and Knowledge Discovery, 2004, pp. 320–330.
[26] L.V.S. Lakshmanan, R. Ng, J. Han, A. Pang, Optimization of constrained frequent set queries with 2-variable constraints, in: ACM SIGMOD Conference, Philadelphia, PA, USA, 1999, pp. 157–168.
[27] B. Lan, B.C. Ooi, K.L. Tan, Efficient indexing structures for mining frequent patterns, in: IEEE International Conference on Data Engineering, 2002, pp. 453–462.
[28] H. Mannila, H. Toivonen, A.I. Verkamo, Efficient algorithm for discovering association rules, in: The AAAI Workshop on Knowledge Discovery in Databases, 1994, pp. 181–192.
[29] H. Mannila, H. Toivonen, On an algorithm for finding all interesting sentences, in: The European Meeting on Cybernetics and Systems Research, vol. II, 1996.
[30] R.T. Ng, L.V.S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained associations Rules, in: ACM SIGMOD Conference, Seattle, WA, USA, 1998, pp. 13–24.
[31] J.S. Park, M.S. Chen, P.S. Yu, Using a hash-based method with transaction trimming for mining association rules, IEEE Transactions on Knowledge and Data Engineering 9 (5) (1997) 812–825.
[32] H. Pinto, J. Han, J. Pei, K. Wang, in: Multi-dimensional sequential pattern mining, in: ACM International Conference on Information and Knowledge Management, 2001, 81–88.
[33] N.L. Sarda, N.V. Srinivas, An adaptive algorithm for incremental mining of association rules, in: IEEE International Workshop on Database and Expert Systems, 1998, pp. 240–245.
[34] A. Savasere, E. Omiecinski, S. Navathe. An efficient algorithm for mining association rules in large databases, in: ACM International Conference on Very Large Data Bases, 1995, pp. 432–444.
[35] S. Thomas, S. Bodagala, K. Alsabti, S. Ranka, in: An efficient algorithm for the incremental update of association rules in large databases, in: The International Conference on Knowledge Discovery and Data Mining, 1997, pp. 263–266.
[36] K. Wang, L. Tang, J. Han, J. Liu, Top down FP-Growth for association rule mining, in: Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, 2002, pp. 334–340.
[37] J. Widom, Research problems in data warehousing, in: ACM International Conference on Information and Knowledge Management, 1995.
[38] X. Wu, S. Zhang, Synthesizing high-frequency rules from different data sources, IEEE Transactions on Knowledge and Data Engineering 15 (2) (2003) 353–367.
[39] S. Zhang, X. Wu, C. Zhang, Multi-database mining, IEEE Computational Intelligence Bulletin 2 (1) (2003) 5–13.
[40] Z. Zheng, R. Kohavi, L. Mason, Real world performance of association rule algorithms, in: The International Conference on Knowledge Discovery and Data Mining, 2001.