The VLDB Journal (2002) 11: 153–175 / Digital Object Identifier (DOI) 10.1007/s00778-002-0069-6

A template model for multidimensional inter-transactional association rules

Ling Feng[1,⋆], **Jeffrey Xu Yu**[2], **Hongjun Lu**[3], **Jiawei Han**[4]

[1] Infolab, Department of Information Systems and Management, Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands; e-mail: ling@kub.nl
[2] Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Shatin, N.T., China; e-mail: yu@se.cuhk.edu.hk
[3] Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, China; e-mail: luhj@cs.ust.hk
[4] Department of Computing Science, University of Illinois at Urbana-Champaign, 1304 West Springfield Ave, IL 61801 USA; e-mail: hanj@cs.uiuc.edu

Edited by M.T. Özsu. Received: February 16, 2001 / Accepted: June 1, 2002
Published online: September 25, 2002 –

Abstract. Multidimensional inter-transactional association rules extend the traditional association rules to describe more general associations among items with multiple properties across transactions. "*After McDonald and Burger King open branches, KFC will open a branch two months later and one mile away*" is an example of such rules. Since the number of potential inter-transactional association rules tends to be extremely large, mining inter-transactional associations poses more challenges on efficient processing than mining traditional intra-transactional associations. In order to make such association rule mining truly practical and computationally tractable, in this study we present a template model to help users declare the interesting *multidimensional inter-transactional associations* to be mined. With the guidance of templates, several optimization techniques, i.e., joining, converging, and speeding, are devised to speed up the discovery of inter-transactional association rules. We show, through a series of experiments on both synthetic and real-life data sets, that these optimization techniques can yield significant performance benefits.

Keywords: Intra-transactional/inter-transactional association rules – Multidimensional context – Template model

1 Introduction

Since the problem of mining association rules was introduced in [1], a large amount of work has been done in various directions, including efficient, Apriori-like mining methods [4, 26,45,40–42,51,59,19], mining generalized, multi-level, or quantitative association rules [48,49,23,21,20,37,28,25,44], association rule mining query languages [36,52], constraint-based rule mining [27,38,50,52,6,24,18], incremental maintenance of discovered association rules [11], parallel and distributed mining [3,22,12], mining correlations and causal structures [9,46,47], cyclic, interesting, and surprising association rule mining [39,43,13,10], mining frequent itemsets with multiple supports [31,55], and so on.

Recently, the problem of mining *multidimensional inter-transactional association rules* was introduced in [33,32]. It extends the scope of mining association rules from traditional *single-dimensional intra-transactional* associations, to *multidimensional inter-transactional* associations. Intra-transactional associations are the associations among items within the *same transaction*, where the notion of the transaction could be the items bought by the *same customer*, the events happening on the *same day*, etc. However, an inter-transactional association describes the association relationships among *different transactions*, such as "*if company A's stock goes up on day 1, B's stock will go down on day 2, but go up on day 4.*" In this case, whether we treat company or day as the unit of transaction, the associated items belong to different transactions. Moreover, such an inter-transactional association can be extended to associate multiple contextual properties (e.g., time, space, temperature, etc.) in the same rule, so that multidimensional inter-transactional associations can be defined and discovered. For example, if a database contains records about the time and location of buildings and facilities of cities under development, we may be able to find 2-dimensional inter-transactional association rules such as "*After McDonald and Burger King open branches, KFC will open a branch two months later and one mile away*", which involves two dimensions - *time* and *space*.

Multidimensional inter-transactional association rules provide a more detailed view of associations among items because they intend to capture more rich contextual information for association relationships. In comparison, the context for traditional intra-transactional association rules is limited to single transaction. Thus, from both a conceptual and algorithmic point of view, traditional intra-transactional asso-

⋆ *Present address:* Database Group, Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands; e-mail: ling@cs.utwente.nl

ciation rules can be viewed as a simple case of multidimensional inter-transactional association rules. By extension of Apriori [4], two kinds of algorithms, named E/EH-Apriori (Extended/Extended Hash-based Apriori) [33,32] and FITI (First-Intra-Then-Inter) [53], were presented for mining inter-transactional association rules from large data sets. Empirical results indicate that with multidimensional inter-transactional association rules, more comprehensive and interesting knowledge can be detected, but this is at the expense of higher computational cost than traditional association rule mining.

In order to make inter-transactional association rule mining truly practical and computationally tractable, in this study we propose a template model to help such rule discovery. Previous work on traditional association rules demonstrated the effectiveness of constraint/query-based association mining [27,38,50,52,36,6,18]. It is applicable to inter-transactional association mining as well, since users may also have certain interesting inter-transactional contexts in mind, from which to do the mining. For example, users may want to know how stock a's rising behavior *today* affects other stocks *next week*. A rule like "*If a goes down, b will go down 243 days later*" most probably cannot inspire much confidence in stock traders.

Hence, one contribution of this paper is to provide users with a set of constructors to specify the interesting *inter-transactional associations*, so that mining can be focused and the cost incurred is proportionate to what the users want and get. Another contribution of the paper is that we develop several optimization techniques, i.e., *joining, converging* and *speeding*, for mining inter-transactional association rules under rule templates. This allows us to significantly reduce the amount of wasted work performed during the mining process. We demonstrate the effectiveness of these techniques through a series of experiments on both synthetic and real-life data sets.

The remainder of the paper is organized as follows. In Sect. 2, we provide a brief review of the basic concepts of multidimensional inter-transactional association rules. A template model for such extended multidimensional inter-transactional association rules is then introduced in Sect. 3. Section 4 examines the template translation phase in detail. Several optimization techniques and algorithms which utilize templates to speed up the discovery of inter-transactional association rules are discussed in Sects. 5 and 6. The experimental evaluation on both synthetic and real-life data sets is presented in Sect. 7. Section 8 reviews some closely related work. Finally, Sect. 9 concludes the paper with a brief discussion of future work.

2 Association among multidimensional transactions

In this section, we provide some background information for multidimensional inter-transactional association rules. We start with a database of multidimensional transactions, and then define multidimensional inter-transactional association rules.

2.1 Database of multidimensional transactions

In the traditional association mining, the database to be mined is organized as a set of records identified by their transaction IDs. Associations mined refer to the relationships among items within a transaction. That is, we only consider the items themselves and ignore other attributes associated with such transactions, such as time, place, and customers. However, it is often the case that, such attributes, or contextual information of transactions, are of main interests. In Table 1, we show a sample database about fast food outlets. Each record contains a list of items, a, b, c, and d representing different types of outlets, such as MacDonald, KFC, Burger King, together with their locations in terms of x and y coordinates, which can be viewed as the number of blocks from a reference point. The graphical representation of the database is shown in Fig. 1. For such a database, we may not only be interested in which outlets are in the same block, but also the outlets in neighboring blocks, which requires us to consider associations among items from different transactions. With such a motivation, we extended the traditional intra-transactional association mining to inter-transactional, multidimensional association mining [33,32].

Table 1. A sample database

ID	x	y	Items
t_1	0	0	{a, b, c}
t_2	1	0	{b}
t_3	2	0	{b, c}
t_4	3	0	{d }
t_5	4	0	{b, c }
t_6	0	1	{b, c}
t_7	1	1	{d}
t_8	2	1	{a}
t_9	3	1	{b}
t_{10}	4	1	{a}
t_{11}	0	2	{a}
t_{12}	1	2	{b}
t_{13}	2	2	{b, c}
t_{14}	3	2	{b}
t_{15}	4	2	{b}
t_{16}	0	3	{b, c}
t_{17}	1	3	{d}
t_{18}	2	3	{a}
t_{19}	3	3	{b, c}
t_{20}	4	3	{a}

In multidimensional association mining, a transaction (in a generic sense) contains two pieces of information, a list of items and the context under which the items are considered. Items in a transactional database could be of any type of objects, or events that are of interest to a particular application, such as shopping items, shops, gas stations, restaurants, best sellers, etc. The context is usually defined by m attributes, $d_1, d_2, \ldots, d_m$, each of which represents a dimension. Typical dimensional attributes include *time, distance, temperature, latitude*, and so on. Note that the dimensional attributes could be of any kind of attributes related to an application. In our above sample database, we have two numerical dimensional attributes, *x* and *y*, representing the number of blocks in two directions with respect to a certain reference point. The context for a stock movement database could be constructed by

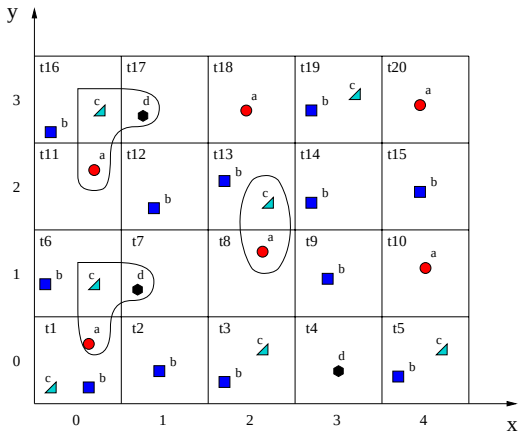


Fig. 1. A graphical representation of the sample database

one dimensional attributes, *trading date*. As another example, a database of best sellers of a chain store, the items are the best selling products, and the dimensional attributes could be the date, or the branch of the store. The diversity of dimensional attributes raises two issues. First, for continuous attributes, items may be associated with different attribute values. For example, the stock price may be different every time unit. In this case, a continuous attribute needs to be discretized into intervals based on the requirement of applications. For example, we can choose a trading day as one interval, a transaction's item list contains all stocks whose prices are up during the day. We can also choose one week as an interval when we are concerned more over long term trend. In this case, the item list contains stocks whose prices are up with respect to the closing prices of last trading day of the week. Second, since we intend to mine inter-transactional associations, transactions have to be ordered. For example, we may have an association rule saying that "*Item A, a best seller at store X, will be a best seller at the next store within two-day's time.*" That is, A will appear in the next transaction in the order of attribute $shop$. While ordering of continuous attributes is easy to determine, ordering nominal attributes seems unnatural from mathematical view point. However, it can be done if we consider the semantics of the attributes and applications. In our best seller example, transactions can either be ordered by the size of shops, or geographical distances between shops. Similarly, the domain of attribute *customer* can be ordered based on *age* or *income* of customers. Although rules mined with different ordering may have different forms, users can prepare the data in such a way that the results will be of most interest to them. Besides, we may not need to have a total order for each dimension. If the relative distance between every two transactions can be identified (neighbor), that is ok. Note that distance is weaker than a total order.

With the above understanding, we assume that the domain of each dimensional attribute is divided into intervals, and transactions are ordered before mining association rules. A database of multidimensional transactions can be defined as follows. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_\omega\}$ denote a set of literals called items, and $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ denote a set of dimensional attributes. A multidimensional transactional database is a set of transactions $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$, where each transaction in $\mathcal{T}$ is in the form of $(d_1,\ d_2,\ \ldots,\ d_m,\ I)$ where $d_i \in D_i, 1 \leq i \leq m$ and $I \in \mathcal{I}$. To emphasize that $(d_1,\ d_2,\ \ldots,\ d_m)$ is the context under which the transaction occurs, we call it the *contextual point* of the transaction, denoted as $\Delta_{(d_1,\ d_2,\ \ldots,\ d_m)}$, and call such a multidimensional transaction an **extended transaction**, and denote it as $\Delta_{(d_1,\ d_2,\ \ldots,\ d_m)}(t)$. Similarly, we can describe the occurrence context of an item $i \in \mathcal{I}$ by associating an m-dimensional contextual point $\Delta_{(d_1,\ d_2,\ \ldots,\ d_m)}$ with i. We call such kind of item an **extended item** and denote it as $\Delta_{(d_1,\ d_2,\ \ldots,\ d_m)}(i)$. Using the notation, the multidimensional transactional database shown in Table 1 can be transformed into the database shown in Table 2.

Note that the contextual point of an extended transaction or an item, is just a point position in the m-dimensional space. Let $n_i = (n_i.d_1,\ n_i.d_2,\ \cdots,\ n_i.d_m)$ and $n_j = (n_j.d_1,\ n_j.d_2,\ \cdots,\ n_j.d_m)$ be two such points, whose values on the m dimensions are represented as $n_i.d_1,\ n_i.d_2,\ \ldots,\ n_i.d_m$ and $n_j.d_1, n_j.d_2, \ldots, n_j.d_m$, respectively. Two points n_i and n_j are equal, if and only if $\forall k (1 \leq k \leq m)\ (n_i.d_k = n_j.d_k)$. A **relative distance** between n_i and n_j is defined as $\Delta\langle n_i, n_j\rangle = (n_j.d_1 - n_i.d_1,\ n_j.d_2 - n_i.d_2, \cdots, n_j.d_m - n_i.d_m)$. Thus, besides the *absolute representation* $(n_i.d_1,\ n_i.d_2,\ \cdots,\ n_i.d_m)$ for point n_i, we can also represent it by indicating its *relative distance* $\Delta\langle n_0, n_i\rangle$ from a certain **reference point** n_0. Let $\mathcal{N} = \{n_1,\ n_2,\ \ldots,\ n_u\}$ be a set of points in an m-dimensional space. The **largest reference point** of $\mathcal{N}$ is the point n_0, where $\forall k\ (1 \leq k \leq m)\ (n_0.d_k = min(n_1.d_k,\ n_2.d_k,\ \ldots,\ n_u.d_k))$.

Example 1 Given two points, $n_1 = (0, 2), n_2 = (1, 1)$, in a two-dimensional space, the largest reference point of $\{n_1, n_2\}$ is $n_0 = (0, 1)$, since $n_0.d_1 = min(n_1.d_1,\ n_2.d_1) = min(0, 1) = 0$ and $n_0.d_2 = min(n_1.d_2,\ n_2.d_2) = min(2, 1) = 1$. □

In the following, we also refer to an m-dimensional point n_i through $\Delta_{(n_i.a_1 - n_0.a_1,\ n_i.a_2 - n_0.a_2,\ \ldots,\ n_i.a_m - n_0.a_m)}$ when the reference point n_0 is clear in the context of discourse.

Given an extended item set, $I_e = \{\Delta_{(d_{1,1},\ldots,d_{1,m})}(i_1), \Delta_{(d_{2,1},\ldots,d_{2,m})}(i_2), \ldots, \Delta_{(d_{k,1},\ldots,d_{k,m})}(i_k)\}$, we call it a **normalized extended item set**, if the reference point of the extended items in I_e is the largest reference point of the set, that is, $\forall j (1 \leq j \leq k)\ \forall i (1 \leq i \leq m)\ (min(d_{j,i}) = 0)$. Similarly, we call an extended transaction set a **normalized extended transaction set**, if all extended item sets of those transactions are normalized extended item sets.

Any non-normalized extended item (transaction) set can be transformed into a normalized one through a **normalization function** called $Norm$, whose intention is to reposition all the involved extended items (transactions) based on the largest reference point of this set. We use $\mathcal{I}_{NE}$ and $\mathcal{T}_{NE}$ to denote the set of all possible normalized extended item sets and normalized extended transaction sets, respectively.

The motivation of having normalized extended item sets can be clearly seen using two such item sets circled in Fig. 1: $I_e = \{\Delta_{(0,0)}(a),\ \Delta_{(0,1)}(c),\ \Delta_{(1,1)}(d)\}$ and $I'_e = \{\Delta_{(0,2)}(a),\ \Delta_{(0,3)}(c),\ \Delta_{(1,3)}(d)\}$. I_e is normalized, and I'_e is not: the minimum value for both x and y for all items in I_e is zero and the minimum y for I'_e is 2. The normalized item set for I'_e is $I''_e = \{\Delta_{(0,0)}(a),\ \Delta_{(0,1)}(c),\ \Delta_{(1,1)}(d)\}$, which

Table 2. A transformed extended transactional database

Transaction	(x, y, I)	Extended transaction	Extended Items
t_1	(0, 0, {a, b, c})	$\Delta_{(0,0)}(t_1)$	$\Delta_{(0,0)}(a),\ \Delta_{(0,0)}(b),\ \Delta_{(0,0)}(c)$
t_2	(1, 0, {b})	$\Delta_{(1,0)}(t_2)$	$\Delta_{(1,0)}(b)$
t_3	(2, 0, {b, c})	$\Delta_{(2,0)}(t_3)$	$\Delta_{(2,0)}(b),\ \Delta_{(2,0)}(c)$
t_4	(3, 0, {d })	$\Delta_{(3,0)}(t_4)$	$\Delta_{(3,0)}(d)$
t_5	(4, 0, {b, c })	$\Delta_{(4,0)}(t_5)$	$\Delta_{(4,0)}(b),\ \Delta_{(4,0)}(c)$
t_6	(0, 1, {b, c})	$\Delta_{(0,1)}(t_6)$	$\Delta_{(0,1)}(b),\ \Delta_{(0,1)}(c)$
t_7	(1, 1, {d})	$\Delta_{(1,1)}(t_7)$	$\Delta_{(1,1)}(d)$
t_8	(2, 1, {a})	$\Delta_{(2,1)}(t_8)$	$\Delta_{(2,1)}(a)$
t_9	(3, 1, {b})	$\Delta_{(3,1)}(t_9)$	$\Delta_{(3,1)}(b)$
t_{10}	(4, 1, {a})	$\Delta_{(4,1)}(t_{10})$	$\Delta_{(4,1)}(a)$
t_{11}	(0, 2, {a})	$\Delta_{(0,2)}(t_{11})$	$\Delta_{(0,2)}(a)$
t_{12}	(1, 2, {b})	$\Delta_{(1,2)}(t_{12})$	$\Delta_{(1,2)}(b)$
t_{13}	(2, 2, {b, c})	$\Delta_{(2,2)}(t_{13})$	$\Delta_{(2,2)}(b),\ \Delta_{(2,2)}(c)$
t_{14}	(3, 2, {b})	$\Delta_{(3,2)}(t_{14})$	$\Delta_{(3,2)}(b)$
t_{15}	(4, 2, {b})	$\Delta_{(4,2)}(t_{15})$	$\Delta_{(4,2)}(b)$
t_{16}	(0, 3, {b, c})	$\Delta_{(0,3)}(t_{16})$	$\Delta_{(0,3)}(b), \Delta_{(0,3)}(c)$
t_{17}	(1, 3, {d})	$\Delta_{(1,3)}(t_{17})$	$\Delta_{(1,3)}(d)$
t_{18}	(2, 3, {a})	$\Delta_{(2,3)}(t_{18})$	$\Delta_{(2,3)}(a)$
t_{19}	(3, 3, {b, c})	$\Delta_{(3,3)}(t_{19})$	$\Delta_{(3,3)}(b),\ \Delta_{(3,3)}(c)$
t_{20}	(4, 3, {a})	$\Delta_{(4,3)}(t_{20})$	$\Delta_{(4,3)}(a)$

is actually the same as I_e. That is, normalized itemset makes it possible to find the same pattern in a multidimensional space.

Property 1 Any superset of a normalized extended item (transaction) set is also a normalized extended item (transaction) set. □

This property can be proven easily from the definition of normalized extended item (transaction) set.

2.2 Multidimensional inter-transactional association rules

From the multidimensional transaction database defined above, we can now define multidimensional inter-transactional association rules.

Definition 1 A multidimensional inter-transactional association rule *is an implication of the form* $X \Rightarrow Y$*, where* $X, Y \subset \mathcal{I}_E$, $X \cup Y \subset \mathcal{I}_{NE}$, *and* $X \cap Y = \emptyset$. □

Different from classical intra-transactional association rules, an inter-transactional association rule provides the occurrence context for associated items by means of a normalized extended item set $X \cup Y$. That is, all items in an association rule use the same reference point. For example, a rule that predicts the stock price movement – "*if stock 'a' increases one day, and stock 'c' increases the following day, then most probably stock 'e' will increase on the fourth day*", can be expressed by an one-dimensional inter-transactional association rule "$\Delta_{(0)}(a),\ \Delta_{(1)}(c) \Rightarrow \Delta_{(3)}(e)$", where 0, 1, and 3 are days from any trading date.

Similar to intra-transactional association rules, we use *support* and *confidence* as two major measurements for multidimensional inter-transactional association rules. Traditionally, the support of a rule $X \Rightarrow Y$ is the fraction of transactions that contains $X \cup Y$ over the whole transactions, and the confidence of the rule is the fraction of transactions containing X that also contain Y. However, to measure multidimensional inter-transactional association rules, which may span different transactions, the traditional support concept must be extended accordingly from the original *single-transaction-based* to *transaction-set-based*.

We first extend the concept of a transaction containing a set of items.

Minimal containment relationship between an extended transaction set and a normalized extended item set. We define that T_e **contains** I_{ne}, if and only if

(a) For $\forall \Delta_{(d_{x,1},\ d_{x,2},\ ...,\ d_{x,m})}(i_x) \in I_{ne}$, there exists an extended transaction $\Delta_{(d_{x,1},\ d_{x,2},\ ...,\ d_{x,m})}(t) \in Norm(T_e)$, where ($i_x \in t$), and
(b) there exists no other extended transaction set T'_e, such that $T'_e \subset T_e$ and (a) holds.

Example 2 Refer to our sample transaction database in Fig. 1 and Table 2. Given $I_{ne} = \{\Delta_{(0,0)}(a),\ \Delta_{(0,1)}(c),\ \Delta_{(1,1)}(d)\}$, two extended transaction sets contain I_{ne}. They are $T_1 = \{\Delta_{(0,0)}(t_1),\ \Delta_{(0,1)}(t_6),\ \Delta_{(1,1)}(t_7)\}$ and $T_2 = \{\Delta_{(0,2)}(t_{11}),\ \Delta_{(0,3)}(t_{16}),\ \Delta_{(1,3)}(t_{17})\}$. After normalization, both of them have transactions at $\Delta_{(0,0)}, \Delta_{(0,1)}$ and $\Delta_{(1,1)}$ with items a, c, and d, respectively. For the same reason, we have three extended transaction sets in the database, i.e., $\{\Delta_{(0,0)}(t_1), \Delta_{(0,1)}(t_6)\}$, $\{\Delta_{(0,2)}(t_{11}),\ \Delta_{(0,3)}(t_{16})\}$ and $\{\Delta_{(2,1)}(t_8),\ \Delta_{(2,2)}(t_{13})\}$ that contain $I'_{ne} = \{\Delta_{(0,0)}(a),\ \Delta_{(0,1)}(c)\}$. □

Now we define support and confidence of a multidimensional inter-transactional association rule.

Definition 2 *Given a multidimensional transaction database with N extended transactions, the* **support** *and* **confidence** *of a multidimensional inter-transactional association rule $X \Rightarrow Y$ is defined as: $support(X \Rightarrow Y) = |T_{xy}|/N|$ and $confidence(X \Rightarrow Y) = |T_{xy}|/|T_x|$, where $|T_{xy}|$ and $|T_x|$ are the number of extended transaction sets that contain $X \cup Y$ and X, respectively, in the database.* □

Example 3 Suppose we have an inter-transactional association rule "$\Delta_{(0,0)}(a),\ \Delta_{(0,1)}(c) \Rightarrow \Delta_{(1,2)}(d)$" from the database in Fig. 1, with $X = \{\Delta_{(0,0)}(a), \Delta_{(0,1)}(c)\}$ and $Y = \{\Delta_{(1,1)}(d)\}$. According to Example 2, the total number of extended transaction sets that contain $X \cup Y$ is 2. Thus, support = $|T_{xy}|/N = 2/20 = 10\%$. Among the 3 extended transaction sets that contain X, only 2 of them contain $X \cup Y$. Thus, $confidence = |T_{xy}|/|T_x| = 2/3 \approx 67\%$. □

Although we extended the traditional association rules to a multidimensional context, the property of Apriori based association rule mining algorithm still holds:

Property 2 Given two normalized extended itemsets X and X' where $X \subset X'$, $support(X) \geq support(X')$. □

Proof. Let T_X and $T_{X'}$ denote a set of extended transaction sets that contain X and X', respectively. It is obvious that $|T_X| \geq |T_{X'}|$, since for any extended transaction set $t_{x'}$ containing X', we can find a corresponding minimal extended transaction set $t_x \subset t_{x'}$ containing X.
Therefore, we have

$$support(X) = \frac{|T_X|}{N} \geq \frac{|T_{X'}|}{N} = support(X').$$ □

As the databases to be mined usually contain a huge amount of data with the fast growing data collection technologies, Property 2 not only enables simplifying the computation of the support level of extended itemsets, but also maintains the important monotonic property that the support of an itemset will not be larger than the support of any of its subsets. We like to have this downward closure property since it is the base of a large set of efficient association rule mining algorithms. All existing algorithms that mine multidimensional inter-transactional association rules use this property [33,32, 53].

3 A template model for inter-transactional association rules

A frequently encountered problem in association rule mining is that mining systems may return quite a large number of rules. With inter-transactional associations which capture more knowledge than intra-transactional ones, the number of rules returned tends to be even more. Thus, from the standpoints of both users and computational costs, it is necessary to restrict the search space and perform human-centered data mining. In this section, we present a template model to enable users to specify what kinds of interesting *multidimensional inter-transactional association rules* are to be mined.

3.1 A two-level template model

Templates are effective ways for users to specify the kind of associations they want to mine from a multidimensional transactional database. Let's use a stock movement database as an example. The database has one dimensional attribute, trading day and lists of stocks whose prices are up on the trading day. An investor may be interested to know *"When two stocks rise together on the same day, which stock will go up one week later?"*. This is equivalent to association rules in the form of $\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(7)}(*)$, where * represents any stock. Using this rule as the template, the mining algorithm can reduce the search space dramatically. For example, it only needs to examine transactions that are 7 days apart. More importantly, the user will be only given those rules that s/he is interested in. While using rules in some particular form as a template can serve the purpose, there are some related issues. For example, if a user would like to know *"When two stocks rise together on the same day, which stock will go up within a week?"*, s/he may need to give a set of such template rules like $\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(1)}(*),\ \Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(2)}(*)$, and etc. Obviously, it is rather ineffective to do so.

With the above observation, we propose a two-level template model for association mining in multidimensional transactional databases. Users specify high-level **template expressions**, and the system transforms them into **template instances**. Let $\mathcal{P}_{oint}$ be the set of all possible contextual points in the transactional database in consideration. We call an extended item with uncertain contextual point and/or item an **extended item variable**, denoted as $\nabla(x)$, where $\nabla \in \mathcal{P}_{oint}$ and $x \in \mathcal{I}$. A **multidimensional inter-transactional association rule template expression** consists of a set of extended item variables that satisfy indicated constraints in the form of $\nabla_1(x_1),\ \nabla_2(x_2),\ \ldots,\ \nabla_p(x_p) \Rightarrow \nabla_{p+1}(x_{p+1}), \ldots, \nabla_{p+q}(x_{p+q}) \mid (\mathcal{C}_{item}, \mathcal{C}_{context})$, where $\nabla_1(x_1), \nabla_2(x_2), \ldots, \nabla_{p+q}(x_{p+q})$ are extended item variables, $\mathcal{C}_{item}$ is a constraint Boolean expression on items $x_1, x_2, \ldots, x_{p+q}$, and $\mathcal{C}_{context}$ is a constraint Boolean expression on item contexts $\nabla_1, \nabla_2, \ldots, \nabla_{p+q}$. All contextual points ∇_i(s) in a rule template should be positioned with respect to a common reference point. The above rule template implies that only association rules satisfying both $\mathcal{C}_{item}$ and $\mathcal{C}_{context}$ are to be detected.

Example 4 "$\nabla_1(x_1), \nabla_2(x_2) \Rightarrow \nabla_3(x_3) \mid (\mathcal{C}_{item}, \mathcal{C}_{context})$", where $\mathcal{C}_{item} : true$ and $\mathcal{C}_{context} : (\nabla_1 = \nabla_2 = \Delta_{(0)}) \wedge (0 < DistPoint(1, \nabla_2, \nabla_3) < 7)$ is a template expression, with three variables. $DistPoint$ is a function which returns the distance between two contextual points along a certain specified dimension (dimension 1 in this example). This template expression specifies all the association rules with two items in the antecedent and one item in the consequent. The contextual point of the two antecedent items are the same and the consequent item's is seven unit apart. In the context of our stock movement example, those association rules answer the query "When two stocks rise together on the same day, which stock will go up within a week?". □

Rule template instances are instantiation of corresponding rule template expressions, where extended item variables are instantiated to actual contextual points.

Example 5 "$\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(1)}(*)$", "$\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(2)}(*)$", $\cdots$, "$\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(7)}(*)$" are seven template instances corresponding to the template expression in the above example. Because no constraints on

x_1, x_2 and x_3, the three item variables are instantiated to any (*). □

As item constraints have been extensively studied in the traditional association rule mining [27,38,50,52,36,6,24,18], we focus on context constraints in this paper. The Boolean expression $\mathcal{C}_{context}$ is in a conjunctive normal form $c_1 \wedge c_2 \wedge \ldots \wedge c_r$, where each c_i $(1 \leq i \leq r)$ is of the form $c_{i,1} \vee c_{i,2} \vee \ldots \vee c_{i,t}$. Each allowed $c_{i,j}$ $(1 \leq i \leq r,\ 1 \leq j \leq t)$ is a constraint defined in the following sections.

With the two-level template model, mining multidimensional inter-transactional association rules consists of four phases: *template translation, mining planning, frequent normalized extended itemset discovery*, and *inter-transactional association rule generation.*

Phase-1 (template translation)

The input of this phase is user-specified high-level template expressions that specify a set of Boolean constraints that extended item variables of the rules should satisfy (as shown in Example 4). Such template expressions are transformed during this phase to produce a set of template instances (as shown in Example 5), with all the contexts of extended item variables being instantiated. A template may imply one or several *template instances* to which the rules discovered later must conform.

The template translation phase interprets and translates template expressions given as input by users into a set of template instances. In a template instance, all the contexts of extended item variables are instantiated so as to provide concrete guidance for the following mining processes. Section 4 examines this template translation procedure in detail.

Phase-2 (mining planning)

Like traditional association rule mining, we first discover all normalized extended itemsets with support not less than a user-specified $minsup$ threshold. We call these itemsets **frequent** normalized extended itemsets. From the frequent itemsets discovered, we then derive inter-transactional association rules with confidence not less than a user-specified $minconf$ threshold.

Different from traditional itemsets where all items are within the same transaction, an extended k-itemset under the circumstance of inter-transactional associations may span several transactions. For example, to get one-dimensional inter-transactional rules
"$\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(7)}(*)$" and
"$\Delta_{(0)}(*), \Delta_{(0)}(*) \Rightarrow \Delta_{(14)}(*)$", we need to identify frequent 3-itemsets by counting all those candidate 3-itemsets $C_3^* = \{\{\Delta_{(0)}(*), \Delta_{(0)}(*), \Delta_{(7)}(*)\}, \{\Delta_{(0)}(*), \Delta_{(0)}(*), \Delta_{(14)}(*)\}\}$ across every 8 and 15 consecutive transactions.[1]

The purpose of this phase is to identify candidate itemsets C_k^* to count at each pass $k(\leq RuleLen)$, and decide the generation plan for candidate $RuleLen$-itemsets. Details for mining plan generation are described in Sect. 5.

[1] In the paper, we use C_k^* to represent the set of candidate itemsets with detailed contextual information, and C_k to represent the set of candidate itemsets with both detailed contexts and item IDs. The same for L_k^* and L_k.

Phase-3 (frequent normalized extended itemset discovery)

In this phase, we find the set of all frequent normalized extended itemsets identified in Phase-2. Two algorithms for generating frequent normalized extended itemsets based on different mining plans are described in Sect. 6.

Phase-4 (inter-transactional association rule generation)

Using the frequent normalized extended itemsets, we can find the desired inter-transactional association rules. The generation of inter-transactional association rules is similar to the generation of classical association rules [4] with minor modifications.

In the following sections, we describe how contextual constraints are defined.

3.2 Contextual constraints in templates

In a dimensional space, in addition to contextual points, we introduce the concept of $scope$. A contextual scope is the subspace delimited by two contextual points $P_s = \Delta_{(d_{s_1}, d_{s_2}, \ldots, d_{s_m})}$ and $P_e = \Delta_{(d_{e_1}, d_{e_2}, \ldots, d_{e_m})}$ where $\forall k\ (1 \leq k \leq m)\ (d_{s_k} \leq d_{e_k})$, denoted as $s = [P_s, P_e]$. A contextual constraint is no more a predicate on contextual points and contextual scopes. It is defined based on a set of context-oriented operators and context-oriented functions.

3.2.1 Context-oriented operations

We first define a set of operators whose operands are either contextual points or contextual spaces. Let p and p' be two contextual points, where $p = \Delta_{(d_1, d_2, \ldots, d_m)}$ and $p' = \Delta_{(d'_1, d'_2, \ldots, d'_m)}$. Let s and s' be two contextual scopes, where $s = [s_s, s_e] = [\Delta_{(d_{s_1}, d_{s_2}, \ldots, d_{s_m})},\ \Delta_{(d_{e_1}, d_{e_2}, \ldots, d_{e_m})}]$ and $s' = [s'_s, s'_e] = [\Delta_{(d'_{s_1}, d'_{s_2}, \ldots, d'_{s_m})},\ \Delta_{(d'_{e_1}, d'_{e_2}, \ldots, d'_{e_m})}]$. We have the following context-oriented operators:

Operators on contextual points

The operators $=, \preceq, \prec$ on two contextual points p and p' are defined as follows:

- $p = p'$, iff $\forall k\ (1 \leq k \leq m)\ (d_k = d'_k)$.
- $p \preceq p'$, iff $\forall k\ (1 \leq k \leq m)\ (d_k \leq d'_k)$.
- $p \prec p'$, iff $\forall k\ (1 \leq k \leq m)\ (d_k \leq d'_k) \wedge \exists k\ (1 \leq k \leq m)\ (d_k < d'_k)$.

Operators on a contextual point and a scope

The operators $inner,\ priori,\ rear$ on a contextual point p and a contextual scope $s = [s_s, s_e]$ are defined as follows:

- $inner(p, s) = true$, iff $s_s \preceq p \preceq s_e$
- $prior(p, s) = true$, iff $p \prec s_s$
- $rear(p, s) = true$, iff $s_e \prec p$

Operators on contextual scopes

The operators $precedent,\ inclusive,\ overlap$ on two contextual scopes $s = [s_s, s_e]$ and $s' = [s'_s, s'_e]$ are defined as follows:

- $precedent(s,\ s') = true$, iff $s_e \prec s'_s$.
- $inclusive(s,\ s') = true$, iff $s_s \preceq s'_s \wedge s'_e \preceq s_e$.
- $overlap(s,\ s') = true$, iff there exists a point $p_i = \Delta_{(d_{i_1}, d_{i_2}, \ldots, d_{i_m})}$, such that $inner(p_i,\ s) \wedge inner(p_i,\ s')$.

3.2.2 Context-oriented functions

Besides the above context-oriented operators, we define the following five functions in an m-dimensional space.

- $Size(k,\ s)$ returns the size of the contextual scope s (i.e., number of dimensional values) on the kth dimension, which is $d_{e_k} - d_{s_k} + 1$.
- $DistPoint(k,\ p,\ p')$ returns the relative distance between the two points p and p' on the kth dimension, which is $d'_k - d_k$.
- $DistScope(k,\ s,\ s')$ returns the relative distance between the two scopes s and s' on the kth dimension, which is $d'_{s_k} - d_{e_k}$. Note that $DistScope$ is only meaningful when the two scopes have precedent relationship.
- $DistPointScope(k,\ p,\ s)$ returns the relative distance between contextual point p and contextual scope s on the kth dimension. It equals to $(d_{s_k} - d_k)$ when $prior(p,\ s) = true$; and $(d_k - d_{e_k})$ when $rear(p,\ s) = true$. Note that $DistPointScope$ is only meaningful when the point is not inside the scope.
- $Intersect(s,\ s')$ returns the overlapped scope between s and s'. It equals to s' when $inclusive(s,\ s') = true$; and $[\Delta_{(d'_{s_1},d'_{s_2},\ldots,d'_{s_m})},\ \Delta_{(d_{e_1},d_{e_2},\ldots,d_{e_m})}]$ when $overlap(s,\ s') = true$. Note that $Intersect$ is only meaningful when the two scopes have either inclusive or overlap relationship.

3.2.3 Contextual constraints

With the above defined context-oriented operators and functions, we can define six classes of contextual constraints as follows:

1. Constant constraints.
   - Point constraint $\nabla = \Delta_{(d_1,d_2,\ldots,d_m)}$, indicating that ∇ is at the point $\Delta_{(d_1,d_2,\ldots,d_m)}$.
   - Scope constraint $inner(\nabla,\ [\Delta_{(d_{s_1},\ldots,d_{s_m})},\ \Delta_{(d_{e_1},\ldots,d_{e_m})}])$, indicating that ∇ lies inside the scope $[\Delta_{(d_{s_1},\ldots,d_{s_m})}, \Delta_{(d_{e_1},\ldots,d_{e_m})}]$.
2. Constraints between two points $\nabla_1, \nabla_2 \in \mathcal{P}_{oint}$.
   - $\nabla_1\ \theta\ \nabla_2$, where θ is one of the operators in $\{=,\ \preceq,\ \prec\}$.
   - $DistPoint(k, \nabla_1, \nabla_2)\ \theta\ v$, where $1 \le k \le m$, v is a nonnegative integer, and θ is one of the operators in $\{=, \neq, <, \le, >, \ge\}$.
3. Constraints on one scope $s \in \mathcal{S}_{cope}$.
   - $Size(k, s)\ \theta\ v$, where $1 \le k \le m$, $s \in \mathcal{S}_{cope}$, v is a positive integer, and θ is one of the operators in $\{=, \neq, <, \le, >, \ge\}$.
4. Constraints between two scopes $s_1, s_2 \in \mathcal{S}_{cope}$.
   - $\theta\ (s_1,\ s_2)$, where θ is one of the operators in $\{precedent,\ inclusive,\ overlap\}$.
   - $DistScope(k, s_1, s_2)\ \theta\ v$, where $1 \le k \le m$, v is a positive integer, and θ is one of the operators in $\{=, \neq, <, \le, >, \ge\}$.
5. Constraints between a point $\nabla \in \mathcal{P}_{oint}$ and a scope $s \in \mathcal{S}_{cope}$.
   - $\theta\ (\nabla,\ s)$, where θ is one of the operators in $\{inner,\ prior,\ rear\}$.
   - $DistPointScope(k, \nabla, s)\ \theta\ v$, where $1 \le k \le m$, v is a positive integer, and θ is one of the operators in $\{=, \neq, <, \le, >, \ge\}$.
6. Aggregate constraints.
   - $agg(X)\ \theta\ v$, where agg is one of the aggregate functions in $\{min, max\}$, X is one of the following context-oriented functions: $DistPoint(k, \nabla_1, \nabla_2)$, $DistScope(k, s_1, s_2)$, $DistPointScope(k, \nabla, s)$, $Size(k, s)$, and v is a nonnegative integer.

Any contextual scope in the above constraints can be obtained and substituted by two other scopes through the $Intersect$ function.

3.3 Examples

We conclude our discussion about the template model with a few example contextual expressions in the context of stock movement databases. This will illustrate the expressive power of our template expressions to a certain extent.

Example 6 "When two stocks rise together on the same day, which stock will go up one or two weeks later?" □

One template for this association can be "$\nabla_1(x_1),\ \nabla_2(x_2)\ \Rightarrow\ \nabla_3(x_3) \mid (\mathcal{C}_{item},\ \mathcal{C}_{context})$", where $\mathcal{C}_{item} : true$ and $\mathcal{C}_{context} : (\nabla_1 = \nabla_2 = \Delta_{(0)})\ \wedge\ (DistPoint(1, \nabla_2, \nabla_3) = 7\ \vee\ DistPoint(1, \nabla_2, \nabla_3) = 14)$.

Example 7 "If stock 'a' rises 1 day, and within the following 2 days another different stock rises, will stock 'a' continuously go up within the next 3 days following the rise of the second stock?" □

One template for this association can be "$\nabla_1(a),\ \nabla_2(x_2)\ \Rightarrow\ \nabla_3(a) \mid (\mathcal{C}_{item},\ \mathcal{C}_{context})$", where $\mathcal{C}_{item} : (x_2 \neq a)$ and $\mathcal{C}_{context} : (\nabla_1 = \Delta_{(0)})\ \wedge\ inner(\nabla_2,\ [\Delta_{(1)},\ \Delta_{(2)}])\ \wedge\ (1 \le DistPoint(1, \nabla_2, \nabla_3) \le 3)$.

Example 8 "If stock 'a' rises 1 day, and directly after that there is a stock rising within a period whose maximal span is 2 days, then which stock will rise during the next period of the same length?" □

One template for this association can be "$\nabla_1(a), \nabla_2(x_2)\ \Rightarrow\ \nabla_3(x_3) \mid (\mathcal{C}_{item},\ \mathcal{C}_{context})$" where $\mathcal{C}_{item} : true$ and $\mathcal{C}_{context} : (\nabla_1 = \Delta_{(0)})\ \wedge\ inner(\nabla_2,\ s)\ \wedge\ DistPointScope(1, \nabla_1, s) = 1\ \wedge\ max(Size(1,\ s)) = 2\ \wedge\ inner(\nabla_3,\ s')\ \wedge\ Size(1,\ s) = Size(1,\ s')\ \wedge\ DistScope(1, s, s') = 1$.

4 Template translation phase

We provide some heuristics for the translation of a template into template instances. It proceeds briefly in the following five steps:

Step 1: unify comparison constraints with equal comparison operator (=)

The comparison constraints $(X\ \theta\ v)$ involving one of the comparison operators in $\{>, \geq <, \leq, \neq\}$ and an integer v are first identified from the template Boolean expression $\mathcal{C}_{context}$. This kind of constraints includes $Size(k, s)\ \theta\ v$, $DistPoint(k, \nabla_1, \nabla_2)\ \theta\ v$, $DistScope(k, s_1, s_2)\ \theta\ v$, and $DistPointScope(k, \nabla, s)\ \theta\ v$, where $\nabla_1, \nabla_2 \in \mathcal{P}_{oint}$ and $s_1, s_2, s \in \mathcal{S}_{cope}$. We then transform each of them $(X\ \theta\ v)$ into a set of disjunctive constraints $(X = w_1) \vee \ldots \vee (X = w_s)$, using only the equal comparison operator (=) plus integers $w_1, w_2, \ldots, w_s$ that satisfy the constrained scope requirement $(\theta\ v)$. For example, the comparison constraints in the template in Example 7 "$(1 \leq DistPoint(1, \nabla_2, \nabla_2) \leq 3)$" can be translated into 3 disjunctive constraints "$DistPoint(1, \nabla_2, \nabla_3) = 1 \vee DistPoint(1, \nabla_2, \nabla_3) = 2 \vee DistPoint(1, \nabla_2, \nabla_3) = 3$."

In case that users only indicate a lower bound or a upper bound of a comparison constraint like "$DistPoint(1, \nabla_2, \nabla_3) > 1$", we can invoke default constraints for X based on the observation that in real-world applications, users are usually interested in associations happening within a certain range, such as gas stations and fast-food outlets within 50 miles, stock indexes rising within a week, etc.. Here, we assume that for an extended itemset to be of interest, there always exists a maximal contextual span along each dimension, i.e., $maxspan_1, maxspan_2, \ldots, maxspan_m$. By default, the following constraints are always valid.

1) $0 \leq DistPoint(k, \nabla_1, \nabla_2) \leq maxspan_k - 1$
2) $1 \leq DistScope(k, s_1, s_2) \leq maxspan_k - 1$
3) $1 \leq DistPointScope(k, \nabla, s) \leq maxspan_k - 1$
4) $1 \leq Size(k, s) \leq maxspan_k$

After Step 1, one comparison constraint $(X\theta v)$ is converted into several constraints with equal comparison operator. At the first glance, Step 1 may result in a significant increase in the size of templates. Nevertheless, considering in quite a few applications, users are interested in association relationships within a certain narrow scope, i.e., the number of valid values regarding X in $(X\ \theta\ v)$ may not be huge, such an increase in the size of templates may not be as significant as expected.

Step 2: unify aggregate constraints with equal comparison operator (=)

Similar to comparison constraints, aggregate constraints $(agg(X)\ \theta\ v)$ involving aggregate operators (max, min) in the template Boolean expression can be transformed into a set of disjunctive constraints using equal comparison operator (=) plus a set of integers. The above default constraints on X (i.e., $Size(k, s)$, $DistPoint(k, \nabla_1, \nabla_2)$, $DistScope(k, s_1, s_2)$, and $DistPointScope(k, \nabla, s)$) can be applied when users bound X with only max or min. For instance, by taking the default constraint 4) into account, the aggregate constraint in Example 8 "$max(Size(1,\ s_1)) = 2$" can be translated into 2 constraints "$Size(1,\ s_1) = 1 \vee Size(1,\ s_1) = 2$".

Step 3: augment the template Boolean constraints for completeness

There exists the situation that users do not indicate any constraint for some ∇_i(s) $(1 \leq i \leq p + q)$ in the template Boolean expression $\mathcal{C}_{context}$. The aim of step 3 is to check and augment a default constant contextual scope constraint

"$inner(\nabla_i,\ [\Delta_{(0,0,\ldots,0)}, \Delta_{(maxspan_1, maxspan_2, \ldots, maxspan_m)}])$"

for these ∇_i(s), based on the fact that each extended itemset is supposed to occur under a maximal contextual scope of interest to applications.

Step 4: transform the template Boolean constraint expression into a disjunctive normal form

For convenience, users declare their context constraint expressions $\mathcal{C}_{context}$(s) in a conjunctive normal form. After Step 3, these template Boolean expressions are like $(c_{1,1} \vee c_{1,2} \vee \ldots \vee c_{1,t_1}) \wedge (c_{2,1} \vee c_{2,2} \vee \ldots \vee c_{2,t_2}) \wedge \ldots \wedge (c_{r,1} \vee c_{r,2} \vee \ldots \vee c_{r,t_r})$.

To facilitate the translation of one template expression into several detailed template instances in the next step, we transform it into an equivalent disjunctive normal form like $(c_{1,1} \wedge c_{2,1} \wedge \ldots \wedge c_{r,1}) \vee (c_{1,2} \wedge c_{2,1} \wedge \ldots \wedge c_{r,1}) \vee \ldots \vee (c_{1,t_1} \wedge c_{2,t_2} \wedge \ldots \wedge c_{r,t_r})$, so that each conjunctive component in the formula can be instantiated independently into a set of template instances in Step 5. For example, the expression "$(\nabla_1 = \nabla_2 = \Delta_{(0)}) \wedge (DistPoint(1, \nabla_2, \nabla_3) = 7 \vee DistPoint(1, \nabla_2, \nabla_3) = 14)$" in Example 6 can be converted into "$(\nabla_1 = \nabla_2 = \Delta_{(0)} \wedge DistPoint(1, \nabla_2, \nabla_3) = 7) \vee (\nabla_1 = \nabla_2 = \Delta_{(0)} \wedge DistPoint(1, \nabla_2, \nabla_3) = 14)$".

Step 5: instantiate extended item variables with constant points

The last step is to instantiate each ∇_i $(1 \leq i \leq p + q)$ in the template rule with a certain constant point based on the given constraints. Such an instantiation process is conducted for each conjunctive constraint component $c_1 \wedge c_2 \wedge \ldots \wedge c_r$ obtained from Step 4. For example, according to the conjunctive constraints in Example 6 "$(\nabla_1 = \nabla_2 = \Delta_{(0)}) \wedge DistPoint(1, \nabla_2, \nabla_3) = 7)$", we can infer that "$\nabla_1 = \nabla_2 = \Delta_{(0)}$" and "$\nabla_3 = \Delta_{(7)}$". Hence, the original template rule "$\nabla_1(x_1), \nabla_2(x_2) \Rightarrow \nabla_3(x_3)$" can be translated into one template instance "$\Delta_{(0)}(x_1), \Delta_{(0)}(x_2) \Rightarrow \Delta_{(7)}(x_3)$". In addition, from the second conjunctive component "$(\nabla_1 = \nabla_2 = \Delta_{(0)} \wedge DistPoint(1, \nabla_2, \nabla_3) = 14)$", we can derive another template instance "$\Delta_{(0)}(x_1), \Delta_{(0)}(x_2) \Rightarrow \Delta_{(14)}(x_3)$" for the template in Example 6.

Example 9 Referring to Example 6, after Step 5, the template "$\nabla_1(x_1),\ \nabla_2(x_2) \Rightarrow \nabla_3(x_3)$" where $\mathcal{C}_{context}$: $(\nabla_1 = \nabla_2 = \Delta_{(0)}) \wedge (DistPoint(1, \nabla_2, \nabla_3) = 7 \vee DistPoint(1, \nabla_2, \nabla_3) = 14)$ can be translated into two template instances: 1) $\Delta_{(0)}(x_1), \Delta_{(0)}(x_2) \Rightarrow \Delta_{(7)}(x_3)$; 2) $\Delta_{(0)}(x_1), \Delta_{(0)}(x_2) \Rightarrow \Delta_{(14)}(x_3)$. □

For a ∇_i bounded with a contextual scope, we first instantiate the scope with a constant contextual scope, and

then assign each point that lies within this scope to ∇_i. Note that the default constraint "$inner(\nabla_i,\ [\Delta_{(0,0,\ldots,0)},\ \Delta_{(maxspan_1,maxspan_2,\ldots,maxspan_m)}])$" always sits behind during this translation step.

Example 10 The constraint "$inner(\nabla_2,\ [\Delta_{(1)}, \Delta_{(2)}])$" in Example 7 is translated into $\nabla_2 = \Delta_{(1)}$ or $\nabla_2 = \Delta_{(2)}$, from which we can derive six template instances for the template "$\nabla_1(a),\ \nabla_2(x_2) \Rightarrow \nabla_3(a)$", where
$\mathcal{C}_{context} : (\nabla_1 = \Delta_{(0)}) \wedge inner(\nabla_2,\ [\Delta_{(1)},\ \Delta_{(2)}]) \wedge (1 \leq DistPoint(1, \nabla_2, \nabla_3) \leq 3)$:
1) $\Delta_{(0)}(a), \Delta_{(1)}(x_2) \Rightarrow \Delta_{(2)}(a)$; 2) $\Delta_{(0)}(a), \Delta_{(1)}(x_2) \Rightarrow \Delta_{(3)}(a)$; 3) $\Delta_{(0)}(a), \Delta_{(1)}(x_2) \Rightarrow \Delta_{(4)}(a)$;
4) $\Delta_{(0)}(a), \Delta_{(2)}(x_2) \Rightarrow \Delta_{(3)}(a)$; 5) $\Delta_{(0)}(a), \Delta_{(2)}(x_2) \Rightarrow \Delta_{(4)}(a)$; 6) $\Delta_{(0)}(a), \Delta_{(2)}(x_2) \Rightarrow \Delta_{(5)}(a)$. □

Example 11 Two contextual scopes s and s' are involved in Example 8. According to the restrictions "$DistPointScope(1, \Delta_{(0)}, s) = 1 \wedge max(Size(1,\ s)) = 2 \wedge Size(1,\ s) = Size(1,\ s') \wedge DistScope(1, s, s') = 1$", we can derive two possible pairs $(s,\ s')$ where $(s = [\Delta_{(1)}, \Delta_{(1)}],\ s' = [\Delta_{(2)}, \Delta_{(2)}])$ and $(s = [\Delta_{(1)}, \Delta_{(2)}],\ s' = [\Delta_{(3)}, \Delta_{(4)}])$. The constraint "$inner(\nabla_2,\ s) \wedge inner(\nabla_3,\ s')$" implies that "$inner(\nabla_2,\ [\Delta_{(1)}, \Delta_{(1)}]) \wedge inner(\nabla_3,\ [\Delta_{(2)}, \Delta_{(2)}])$", or "$inner(\nabla_2,\ [\Delta_{(1)}, \Delta_{(2)}]) \wedge inner(\nabla_3,\ [\Delta_{(3)}, \Delta_{(4)}])$."

From each of them, we can further instantiate ∇_2 and ∇_3 by enumerating all possible constant points within the corresponding scope, and obtain the following five template instances for the template in Example 8
"$\nabla_1(a), \nabla_2(x_2) \Rightarrow \nabla_3(x_3)$":
1) $\Delta_{(0)}(a), \Delta_{(1)}(x_2) \Rightarrow \Delta_{(2)}(x_3)$; 2) $\Delta_{(0)}(a), \Delta_{(1)}(x_2) \Rightarrow \Delta_{(3)}(x_3)$; 3) $\Delta_{(0)}(a), \Delta_{(1)}(x_2) \Rightarrow \Delta_{(4)}(x_3)$; 4) $\Delta_{(0)}(a), \Delta_{(2)}(x_2) \Rightarrow \Delta_{(3)}(x_3)$; 5) $\Delta_{(0)}(a), \Delta_{(2)}(x_2) \Rightarrow \Delta_{(4)}(x_3)$. □

Property 3 All template instances implied by a template can be completely generated by the Template Translation Phase. □

Proof. Steps 1–3 perform semantically equivalent template translation. The simple syntax conversion of the template Boolean constraint expression into an equivalent disjunctive normal form, performed in Step 4, also does not alter the semantics of the template constraint obtained after Step 3, making the final template instantiation in Step 5 complete. □

5 Mining planning phase

Mining multidimensional inter-transactional association rules is a computationally intensive problem, requiring considerable search efforts compared to the classical association rule mining. Template instances specify the patterns of inter-transactional association rules. We need to find all rules that satisfy the template (not a part of template). It is goal-driven mining. Planning is needed. There are two basic planning strategies: separate mining plan and joint mining plan. The separate mining plan is to deal with a single template instance at a time. The basic idea of separate mining is to include all necessary extended j-itemsets in the plan for finding an extended k-itemset template instance ($j < k$). It is based on the observation: the more extended j-itemsets are included in the plan, the more opportunities the uninteresting rules can be pruned in an early stage. The joint mining plan is to deal with multiple template instances together. Different from separate mining plan, it attempts to approach the goals as quickly as possible. In other words, it is based on the observation that spending time for intermediate extended itemsets is unnecessarily large. In order to approach the goal, three techniques are proposed: joining, converging and speeding. Joining technique is to join two arbitrary k_1-itemset and k_2-itemset and form a possible up to (k_1+k_2-1)-itemset. The idea of converging is to find possible shortest paths for all template instances by further utilizing the joining technique. The speeding technique is to identify the starting time of adopting joining and converging.

We illustrate our techniques using 1-dimensional inter-transactional association mining in this paper. Our techniques are applicable to n-dimensional inter-transactional association mining. To simplify expressions, we remove bracket ( ) which surrounds coordinates of contextual points so as to use Δ_i for $\Delta_{(i)}$. In addition, $itemset$ and *extended itemset* can be used interchangeably in the following discussion.

5.1 Separate mining plan

One simple mining plan is to treat each template instance belonging to one rule template separately, and identify candidates C_k^* ($1 \leq k \leq RuleLen$) for each individual template instance in a similar way as Apriori-Gen does [4]. That is, each candidate k-itemset $X'' = \{\Delta_{d_1}(x_1), \Delta_{d_2}(x_2), \cdots, \Delta_{d_{k-1}}(x_{k-1}),\ \Delta_{d_k}(x_k)\}$ in C_k is generated by joining two frequent (k-1)-itemsets $X, X' \in L_{k-1}$, where $X = \{\Delta_{d_1}(x_1),\ \Delta_{d_2}(x_2), \cdots, \Delta_{d_{k-2}}(x_{k-2}),\ \Delta_{d_{k-1}}(x_{k-1})\}$ and $X' = \{\Delta_{d_1}(x_1), \Delta_{d_2}(x_2), \cdots,\ \Delta_{d_{k-2}}(x_{k-2}), \Delta_{d_k}(x_k)\}$, whose first ($k$-2) extended items are the same, and "$(x_{k-1} < x_k) \vee precedent(\Delta_{d_{k-1}},\ \Delta_{d_k})$". Note that all candidate and frequent itemsets under study are normalized by default. Due to the monotonicity property that "*any subset of a frequent itemset must be frequent*", L_{k-1} also includes all the normalized subsets of X'', i.e., $Norm(\{\Delta_{d_2}(x_2),\ \Delta_{d_3}(x_3), \cdots,\ \Delta_{d_{k-1}}(x_{k-1}), \Delta_{d_k}(x_k)\})$, $Norm(\{\Delta_{d_1}(x_1), \Delta_{d_3}(x_3), \cdots, \Delta_{d_{k-1}}(x_{k-1}), \Delta_{d_k}(x_k)\})$, $\cdots$, $Norm(\{\Delta_{d_1}(x_1), \cdots, \Delta_{d_{k-3}}(x_{k-3}), \Delta_{d_{k-1}}(x_{k-1}), \Delta_{d_k}(x_k)\})$ for pruning purpose. For each template instance, candidate itemsets C_k^* are designated from $k = RuleLen$ to 1 as above, at which the mining process will target later on. Since such a plan deals with a template instance separately, we refer to it as the *separate* mining plan.

Example 12 Suppose we have two template instances after template translation: "$\Delta_0(*), \Delta_2(*) \Rightarrow \Delta_4(*), \Delta_6(*)$" and "$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_3(*), \Delta_5(*)$". Table 3 illustrates candidate itemsets identified by the *separate* method. To detect rules conforming to the first template instance, we need to calculate the supports of all candidate 4-itemsets $\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$ in C_4^* to get L_4^*. Before that, two such kinds of candidate 3-itemsets as $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ and $\{\Delta_0(*), \Delta_2(*), \Delta_6(*)\}$ should

Table 3. The *separate* mining plan: candidate itemsets to be counted. Note $W\text{-}Len$ gives the length of the minimal window that covers C_i^* in the 1-dimensional space

$\Delta_0(*), \Delta_2(*) \Rightarrow \Delta_4(*), \Delta_6(*)$			
$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$
$\{\Delta_2(*)\}$	$\{\Delta_0(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_6(*)\}$	
$\{\Delta_4(*)\}$	$\{\Delta_0(*), \Delta_6(*)\}$	$\{\Delta_0(*), \Delta_4(*), \Delta_6(*)\}$	
$\{\Delta_6(*)\}$			
$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_3(*), \Delta_5(*)$			
$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*), \Delta_5(*)\}$
$\{\Delta_1(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_5(*)\}$	
$\{\Delta_3(*)\}$	$\{\Delta_0(*), \Delta_3(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$	
$\{\Delta_5(*)\}$	$\{\Delta_0(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_3(*), \Delta_5(*)\}$	
	$\{\Delta_0(*), \Delta_5(*)\}$		
C_1^*	C_2^*	C_3^*	C_4^*
$\lvert C_1^*\rvert = 7$	$\lvert C_2^*\rvert = 6$	$\lvert C_3^*\rvert = 6$	$\lvert C_4^*\rvert = 2$
$W\text{-}Len(C_1^*) = 7$	$W\text{-}Len(C_2^*) = 7$	$W\text{-}Len(C_3^*) = 7$	$W\text{-}Len(C_4^*) = 7$

be counted in order to generate candidate 4-itemsets. For pruning purpose, another two subsets of $\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$ after normalization, i.e., $Norm(\{\Delta_0(*), \Delta_4(*), \Delta_6(*)\}) = \{\Delta_0(*), \Delta_4(*), \Delta_6(*)\}$ and $Norm(\{\Delta_2(*), \Delta_4(*), \Delta_6(*)\}) = \{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$, will also go to C_3^*. Candidate itemsets under the second template instance are decided in a similar way. □

5.2 Joint mining plan

Counting each candidate itemset in Table 3 requires searching several transactions each time. One question we pose is that "*can we just count those necessary candidate itemsets which span as few transactions as possible ?*" In this section, we discuss several techniques, namely, *joining, converging* and *speeding*, based on the observation that "*reducing the length of the minimal window that covers candidate extended itemsets can substantially reduce the running time of frequent itemsets detection.*"

In inter-transactional associations, both items and their occurrence contexts are captured within an itemset. For example, from two itemsets $U = \{\Delta_0(*), \Delta_2(*), \Delta_3(*)\}$ and $V = \{\Delta_0(*), \Delta_1(*), \Delta_7(*), \Delta_{11}(*)\}$, we know that the last two items of U appear in two consecutive transactions, similar to the first two items in V. Thus, besides the traditional way of joining two (k-1)-itemsets based on the first k-2 common extended items, such common relative positions also offer another possibility for joining two itemsets which could be of different size. In the following, we define the **joinable** condition for two extended itemsets and give a **join** operator.

Definition 3 *Given two itemsets* $U = \{\Delta_{d_{u_1}}(u_1), \Delta_{d_{u_2}}(u_2), \cdots, \Delta_{d_{u_s}}(u_s)\}$ *and* $V = \{\Delta_{d_{v_1}}(v_1), \Delta_{d_{v_2}}(v_2), \cdots, \Delta_{d_{v_t}}(v_t)\}$, *$V$ and U are* **joinable** *if and only if there exist $U' \subset U$ and $V' \subset V$ that satisfy the following two conditions:*

1) *U' and V' has equal number of extended items, denoted as $|U'| = |V'|$.*

Table 4. A join example between U and V

U	$\Delta_0(a)$	$\Delta_2(b)$	$\Delta_4(c)$	
V		$\Delta_0(b)$	$\Delta_2(c)$	$\Delta_4(d)$
$U \oplus V$	$\Delta_0(a)$	$\Delta_2(b)$	$\Delta_4(c)$	$\Delta_6(d)$

2) *There exists a nonnegative integer d, such that for any $\Delta_{d_{u_i}}(u_i) \in U'$, there exists a $\Delta_{d_{v_i}}(v_i) \in V'$, where $(u_i = v_i)$ and $(d_{u_i} = d_{v_i} + d)$. We call d the* **joinable distance**. □

Example 13 For two itemsets $V_1 = \{\Delta_0(a), \underline{\Delta_2(b)}, \Delta_3(c), \underline{\Delta_6(d)}\}$ and $U_1 = \{\underline{\Delta_0(b)}, \underline{\Delta_4(d)}, \Delta_9(e)\}$, there exist a $V_1' = \{\underline{\Delta_2(b)}, \underline{\Delta_6(d)}\}$ and a $U_1' = \{\Delta_0(b), \Delta_4(d)\}$ satisfying the above two defined conditions. Therefore, we say V_1 and U_1 are joinable, and the joinable distance is 2. Comparatively, $V_2 = \{\Delta_0(a), \Delta_2(b), \Delta_4(c)\}$ and $U_2 = \{\Delta_0(d), \Delta_3(d)\}$ are not joinable, as there exist no $V_2' \subset V_2$ and $U_2' \subset U_2$ that satisfy the above two conditions. □

Definition 4 *Assume that* $U = \{\Delta_{d_{u_1}}(u_1), \Delta_{d_{u_2}}(u_2), \cdots, \Delta_{d_{u_s}}(u_s)\}$ *and* $V = \{\Delta_{d_{v_1}}(v_1), \Delta_{d_{v_2}}(v_2), \cdots, \Delta_{d_{v_t}}(v_t)\}$ *are joinable on $U' \subset U$ and $V' \subset V$, with d as the joinable distance. U* **join** *V is given as $(U \ {}_{U'} \oplus_{V'} \ V) = U \cup W$, where $W = \{\Delta_{d_{v_i}+d}(v_i) \mid (\Delta_{d_{v_i}}(v_i) \in V) \wedge (\Delta_{d_{v_i}}(v_i) \notin V')\}$.* □

Example 14 Let $U = \{\Delta_0(a), \underline{\Delta_2(b)}, \underline{\Delta_4(c)}\}$ and $V = \{\underline{\Delta_0(b)}, \underline{\Delta_2(c)}, \Delta_4(d)\}$. The result of U join V on U' and V' where $U' = \{\Delta_2(b), \Delta_4(c)\}$ and $V' = \{\Delta_0(b), \Delta_2(c)\}$ is $\{\Delta_0(a), \Delta_2(b), \Delta_4(c), \Delta_6(d)\}$, as shown in Table 4. The joinable distance is 2.
Let $U_1 = \{\Delta_0(a), \underline{\Delta_1(b)}\}$ and $V_1 = V = \{\underline{\Delta_0(b)}, \Delta_2(c), \Delta_4(d)\}$. The result of U_1 join V_1 on $U_1' = \{\Delta_1(b)\}$ and $V_1' = \{\Delta_0(b)\}$ is $\{\Delta_0(a), \Delta_1(b), \Delta_3(c), \Delta_5(d)\}$. The joinable distance is 1. □

Along with the **joining** operation, the next question arises: "*which itemsets are suitable to be* joined *in order to gen-*

erate candidate $RuleLen$-itemsets which need less counting effort?" One technique we use is called **converging**, which converges all template instances together. To illustrate, let us return to the previous two template instances "$\Delta_0(*), \Delta_2(*) \Rightarrow \Delta_4(*), \Delta_6(*)$" and "$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_3(*), \Delta_5(*)$". We note that some common relative positions exist either within or among template instances. Like $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$, $\{\Delta_2(*), \Delta_4(*), \Delta_6(*)\}$ in the former, and $\{\Delta_1(*), \Delta_3(*), \Delta_5(*)\}$ in the latter, they actually convey the same contextual information (i.e, for every alternate transaction). Based on such observation, we employ three heuristics to help identify those joinable itemsets: (a) appearing in template instances as frequently as possible; (b) with a joinable size as large as possible; and (c) with a window that covers joinable items as small as possible.

Table 5 shows the *joining* mining plan, which derives the target C_4^* by $\{\Delta_0(*), \Delta_1(*)\}$ join $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ as given in Example 14. For $k < 4$, the way to generate C_k^* is similar to that in the *separate* planning. Candidate 3-itemset $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ can be obtained by joining two frequent 2-itemsets $\{\Delta_0(*), \Delta_2(*)\}$ and $\{\Delta_0(*), \Delta_4(*)\}$, which are thus included in C_2^*. Accordingly, to get C_2^*, we need to first count and detect frequent 1-itemsets $\{\Delta_0(*)\}, \{\Delta_1(*)\}, \{\Delta_2(*)\}$ and $\{\Delta_4(*)\}$.

Compared to the *separate* mining plan in Table 3, the *joining* mining plan exhibits a much smaller search space in terms of both candidate numbers and the minimal window that covers C_k^* at each iteration. Especially for the reduction of window size (i.e., starting from each database transaction, each time the number of transactions to look at), our previous experience indicates that the window size plays a dominant role on inter-transactional association rule mining performance [32].

The other technique we use is called **speeding** which aims at limiting the number of database scans in the presence of long rules. We restrict $1 \leq k \leq KLimit$ by performing a series of *joining* operations instead of one. For instance, in order to derive target candidate $\{\Delta_0(*), \Delta_1(*), \Delta_2(*), \Delta_3(*), \Delta_3(*)\}$, we can perform two join operations. First, let $U_1 = V_1 = \{\Delta_0(*), \Delta_1(*)\}$. Then, $U_1 \ {}_{U_1'}\oplus_{V_1'} V_1 = \{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$ when $U_1' = \{\Delta_1(*)\}$ and $V_1' = \{\Delta_0(*)\}$. Second, let U_2 be the result of $U_1 \ {}_{U_1'}\oplus_{V_1'} V_1$, and let $V_2 = \{\Delta_0(*), \Delta_1(*), \Delta_1(*)\}$ obtained by joining two $\{\Delta_0(*), \Delta_1(*)\}$ as the $separate$ planning does. Then, $U_2 \ {}_{U_2'}\oplus_{V_2'} V_2 = \{\Delta_0(*), \Delta_1(*), \Delta_2(*), \Delta_3(*), \Delta_3(*)\}$ when $U_2' = \{\Delta_2(*)\}$ and $V_2' = \{\Delta_0(*)\}$. Thus, after the 3^{rd} pass, we can directly construct C_5^* without L_4^*.

Here, the $KLimit$ parameter can be set either beforehand, or dynamically determined during the mining process. Basically, the number of frequent itemsets returned from previous passes, the number of candidate $RuleLen$-itemsets derived by joining the discovered L_is ($i \leq RuleLen$), and the trade-off between database scanning and extra candidate itemsets counting, etc. are some factors which affect such a parameter setting. We leave this issue to a further study.

The *joining*, *converging* and *speeding* techniques lead us to the *joint* mining plan. Figure 2 outlines such a plan construction procedure. It has two steps. The first Initialization Phase intends to find appropriate joining itemsets for each template instance t_i using the above three heuristics. That is, it first selects an itemset from $CrossCommon$ which registers k_s-itemsets with common relative positions among more than one template instance. If this fails, it selects an itemset from $SelfCommon_i$ which registers k_s-itemsets with common relative positions within t_i itself. If this fails again, then a k_s-itemset with the minimal window size in t_i is chosen (line 5-8). The selected itemset, as well as its subsets used for this itemset's generation and pruning purpose, are stored in a candidate pool (line 9-10). The second Sufficient-Candidate-Pool-Checking Phase examines whether the itemsets in CandidatePool identified at Phase 1 can be successfully joined for each template instance (line 12-13). If not, it continues to look for joinable itemsets following the above three heuristics (line 14) and stores the selected itemset and its subsets in the candidate pool (line 15-16).

Taking the two template instances
$t_1: \Delta_0(*), \Delta_2(*) \Rightarrow \Delta_4(*), \Delta_6(*)$ and
$t_2: \Delta_0(*), \Delta_1(*) \Rightarrow \Delta_3(*), \Delta_5(*)$
in Table 5 for example, let $KLimit$=3, k_s=3, we have $CrossCommon = \{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$,
$SelfCommon_1 = \{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$, and
$SelfCommon_2 = \emptyset$. Figure 2 first selects a 3-itemset $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ from $CrossCommon$ for both t_1 and t_2, and puts it into $CandidatePool$. All its low-level subsets enter $CandidatePool$ as well, making
$CandidatePool = \{\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}, \{\Delta_0(*), \Delta_2(*)\}, \{\Delta_0(*), \Delta_4(*)\}, \{\Delta_0(*)\}, \{\Delta_2(*)\}, \{\Delta_4(*)\}\}$. Later in the Sufficient-Candidate-Pool-Checking Phase, the algorithm checks to ascertain that t_1 can be derived by joining two $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ in $CandidatePool$. In other words, $Join(t_1, CandidatePool) \neq Fail$. However, with all the itemsets in $CandidatePool$ so far, no two itemsets can be joined to derive t_2, that is,
$Join(t_2, CandidatePool) = Fail$. In this case, another 2-itemset $\{\Delta_0(*), \Delta_1(*)\}$ with the minimal window size is selected for t_2, enlarging $CandidatePool$ with $\{\Delta_0(*), \Delta_1(*)\}$ and $\{\Delta_1(*)\}$, i.e., $CandidatePool = \{\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}, \{\Delta_0(*), \Delta_2(*)\}, \{\Delta_0(*), \Delta_4(*)\}, \{\Delta_0(*), \Delta_1(*)\}, \{\Delta_0(*)\}, \{\Delta_2(*)\}, \{\Delta_4(*)\}, \{\Delta_1(*)\}\}$. As two itemsets from $CandidatePool$, i.e., $\{\Delta_0(*), \Delta_1(*)\}$ and $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$, can be joined to generate t_2 (Example 14), $Join(t_2, CandidatePool) \neq Fail$. The mining plan construction thus finishes, with the result shown in Table 5.

6 Frequent normalized extended itemset discovery phase

Based on the two different mining plans, we present two different frequent normalized extended itemset discovery algorithms, namely *separate* and *joint* algorithms. Like Apriori [4], both algorithms make multiple passes over the database. Each pass consists of two phases. First, the set of candidate itemsets C_k as indicated by the corresponding mining plan is generated. The algorithms then scan the database. For each minimal extended transaction set, they determine which candidates in C_k are contained and increment their counts. At the end of the pass, C_k is examined to check which of the candidates are actually frequent, yielding L_k. Considering that a huge number of itemsets in C_2 may be generated, especially in the case of inter-transactional association rules, we adopt a similar technique of hashing as [40] to filter out unnecessary candidate 2-itemsets.

Table 5. The *joint* mining plan: candidate itemsets to be counted

$\Delta_0(*), \Delta_2(*) \Rightarrow \Delta_4(*), \Delta_6(*)$			
$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$
$\{\Delta_2(*)\}$	$\{\Delta_0(*), \Delta_4(*)\}$		
$\{\Delta_4(*)\}$			
$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_3(*), \Delta_5(*)$			
$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*), \Delta_5(*)\}$
$\{\Delta_1(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$		
$\{\Delta_2(*)\}$	$\{\Delta_0(*), \Delta_4(*)\}$		
$\{\Delta_4(*)\}$			
C_1^*	C_2^*	C_3^*	C_4^*
$\|C_1^*\| = 4$	$\|C_2^*\| = 3$	$\|C_3^*\| = 1$	$\|C_4^*\| = 2$
$W\text{-}Len(C_1^*) = 5$	$W\text{-}Len(C_2^*) = 5$	$W\text{-}Len(C_3^*) = 5$	$W\text{-}Len(C_4^*) = 7$

Input: a set of template instances $\{t_1, t_2, \ldots, t_r\}$.
Output: a candidate pool containing candidate itemsets C_k^* ($k = 1, 2, \ldots, k_s, RuleLen$).

Let $SelfCommon_i$ be the set of k_s-itemsets with common relative positions *within* t_i;
Let $CrossCommon$ be the set of k_s-itemsets with common relative positions *among* all t_is.

Initialization Phase
1 $CandidatePool = \emptyset$;
2 **if** ($RuleLen - 1 < KLimit$) **then**
3 $k_s = RuleLen - 1$
4 **else** $k_s = KLimit$; // speeding to reduce the number of database scans
5 **foreach** template instance t_i **do**
6 **if** $Select(t_i, k_s, CrossCommon)$ = Fail **then** // select a k_s-itemset for the joining phase
7 **if** $Select(t_i, k_s, SelfCommon_i)$ = Fail **then**
8 get a k_s-itemset with the minimal window size from t_i;
9 $CandidatePool \leftarrow$ the selected k_s-itemset;
10 $CandidatePool \leftarrow$ all low-level subsets of the selected k_s-itemset;
11 **endfor**

Sufficient-Candidate-Pool-Checking Phase
12 **foreach** template instance t_i **do**
13 **while** $Join(t_i, CandidatePool)$ = Fail **do**
14 select another k-itemset for t_i ($k < k_s$); // following step 6-8
15 $CandidatePool \leftarrow$ the selected k_s-itemset;
16 $CandidatePool \leftarrow$ all low-level subsets of the selected k_s-itemset;
17 **endwhile**
18 **endfor**

Fig. 2. The joint plan generation procedure

6.1 The separate mining algorithm

Figure 6.1 gives the *separate* mining procedure by extension of Apriori.

Pass 1. The candidate set C_1 is generated by attaching each item in $\mathcal{I}$ with all possible contextual positions specified in C_1^* (line 1), i.e., $C_1 = \{\{\Delta_d(i)\} \mid i \in \mathcal{I} \wedge \{\Delta_d(*)\} \in C_1^*\}$. Then, starting from each extended transaction $\Delta_c(t)$, the algorithm checks whether an item i exists in the extended transaction $\Delta_{c+d}(t')$. If so, the count of candidate itemset $\{\Delta_d(i)\}$ increases by one (line 3-5). In addition to counting candidate 1-itemsets, the first pass also has the responsibility of hashing all normalized 2-itemsets (e.g., $\{\Delta_0(i_1), \Delta_{d_2}(i_2)\}$ where $\{\Delta_0(*), \Delta_{d_2}(*)\} \in C_2^*$), which are contained in the current minimal extended transaction set, into a hash table (line 6-8). After the first database scan, the frequent set L_1 is delivered (line 10).

Pass 2. A candidate 2-itemset $\{\Delta_0(i_1), \Delta_{d_2}(i_2)\}$ is generated from any two frequent 1-itemsets $\{\Delta_0(i_1)\}, \{\Delta_{d_2}(i_2)\}$ in L_1 (line 11), i.e., $C_2 = \{\{\Delta_0(i_1), \Delta_{d_2}(i_2)\} \mid \{\Delta_0(*), \Delta_{d_2}(*)\} \in C_2^* \wedge \{\Delta_0(i_1)\} \in L_1 \wedge \{\Delta_{d_2}(i_2)\} \in L_1 \wedge (i_1 < i_2 \vee d_2 > 0) \wedge HashTable[func(\{\Delta_0(i_1), \Delta_{d_2}(i_2)\})]/|\mathcal{T}_E| \geq minsup\}$.

Note that all candidate itemsets should be normalized. For candidate 2-itemsets, their hash values should not be less than $minsup * |\mathcal{T}_E|$. After generating C_2, the algorithm examines every minimal extended transaction set in the database, say $\{\Delta_c(t), \Delta_{c+d_2}(t')\}$, that may contain candidate 2-itemsets like $\{\Delta_0(i_1), \Delta_{d_2}(i_2)\}$. If $(i_1 \in t)$ and $(i_2 \in t')$, the count of 2-itemset $\{\Delta_0(i_1), \Delta_{d_2}(i_2)\}$ will be added by one (line 12-

Table 6. A sample realistic stock data, where a represents IBM, b represents IPG(International Public Group), c represents AES(The AES Corp.), d represents AEE(American Corp.), and e represents XEL(Excel Energy)

Extended trans.	Date	Stocks
$\Delta_{(0)}(t_1)$	day_1	IBM
$\Delta_{(1)}(t_2)$	day_2	IPG, AEE, XEL
$\Delta_{(2)}(t_3)$	day_3	IBM, AES, AEE, XEL
$\Delta_{(3)}(t_4)$	day_4	IBM
$\Delta_{(4)}(t_5)$	day_5	IBM, IPG, AES, AEE, XEL
$\Delta_{(5)}(t_6)$	day_6	IBM, AES, AEE
$\Delta_{(6)}(t_7)$	day_7	IBM, IPG, AEE
$\Delta_{(7)}(t_8)$	day_8	IPG, AES, AEE, XEL
$\Delta_{(8)}(t_9)$	day_9	IPG, AES, AEE, XEL
$\Delta_{(9)}(t_{10})$	day_{10}	IBM

Table 7. The *separate* and *joint* mining plans for $\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_2(*)$

	$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_2(*)$		
separate mining plan	$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$
	$\{\Delta_1(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$	
	$\{\Delta_2(*)\}$		
joint mining plan	$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$
	$\{\Delta_1(*)\}$		
	C_1^*	C_2^*	C_3^*

14). The second scan of the database will deliver the frequent set L_2 (line 15).

Pass k ($1 \leq k \leq RuleLen$). Given L_{k-1}, the candidate generation function *E-Apriori-Gen*(L_{k-1}) returns a superset of L_k (line 17). The procedure has two parts. In the *join* phase, two extended (k-1)-itemsets $X, X' \in L_{k-1}$, which have the first (k-2) extended items in common, are joined to derive a candidate k-itemset. Let $X = \{\Delta_{(d_1)}(x_1), \ldots, \Delta_{(d_{k-2})}(x_{k-2}), \Delta_{(d_{k-1})}(x_{k-1})\}$ and $X' = \{\Delta_{(d_1)}(x_1), \ldots, \Delta_{(d_{k-2})}(x_{k-2}), \Delta_{(d_k)}(x_k)\}$, where $(x_{k-1} < x_k)$ or $precedent(\Delta_{d_{k-1}}, \Delta_{d_k})$. We can generate a candidate k-itemset $X'' = \{\Delta_{(d_1)}(x_1), \ldots, \Delta_{(d_{k-2})}(x_{k-2}), \Delta_{(d_{k-1})}(x_{k-1}), \Delta_{(d_k)}(x_k)\}$. It is obvious that X'' is also normalized due to Property 1 that any superset (X'') of a normalized itemset (X, X') is also a normalized itemset. All k-itemsets obtained in the join phase comprise a set C_k^{join}. Next, in the *prune* phase, all those extended k-itemset(s) in C_k^{join} which have some (k-1)-subset(s) not in L_{k-1} are discarded, leading to the candidate set $C_k = \{X'' \mid (X'' \in C_k^{join}) \wedge (\forall Y \subset X'' (|Y| = k-1) \rightarrow (Y \in L_{k-1}))\}$.

The candidate k-itemsets are counted by making one pass over the database in a similar fashion as counting C_2 (line 18-20), from which L_k is derived (line 21). Such a candidate generation and counting process terminates after some iteration when $k > RuleLen$ or $L_{k-1} = \emptyset$.

In the following, we illustrate the steps of the separate mining algorithm using a small realistic stock data shown in Table 6. The database records 10 days' rising movement of five different stocks in March 2001 in USA, with time as the dimensional attribute. Each transaction details stocks whose prices are lower than the previous trading day. Assume the template instance used is $\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_2(*)$. Table 7 gives the separate mining plan, i.e., candidate itemsets to be counted. Let the support count threshold be 5.

Example 15 According to the mining plan, the set of candidate itemsets at pass 1 is $C_1^* = \{\{\Delta_0(*)\}, \{\Delta_1(*)\}, \{\Delta_2(*)\}\}$. By scanning the database once, we get 13 frequent 1-itemsets with count values:
$count(\{\Delta_0(IBM)\}) = 7$, $count(\{\Delta_0(IPG)\}) = 5$,
$count(\{\Delta_0(AES)\}) = 5$, $count(\{\Delta_0(AEE)\}) = 7$,
$count(\{\Delta_0(XEL)\}) = 5$; $count(\{\Delta_1(IBM)\}) = 6$,
$count(\{\Delta_1(IPG)\}) = 5$, $count(\{\Delta_1(AES)\}) = 5$,
$count(\{\Delta_1(AEE)\}) = 7$, $count(\{\Delta_1(XEL)\}) = 5$;
$count(\{\Delta_2(IBM)\}) = 6$, $count(\{\Delta_2(AES)\}) = 5$,
$count(\{\Delta_2(AEE)\}) = 6$.
From L_1, candidate 2-itemsets $C_2^* = \{\{\Delta_0(*), \Delta_1(*)\}, \{\Delta_0(*), \Delta_2(*)\}\}$ are generated, with 5 frequent itemsets returned at pass 2, which are
$count(\{\Delta_0(IBM), \Delta_1(AEE)\}) = 5$,
$count(\{\Delta_0(AEE), \Delta_1(IBM)\}) = 5$,
$count(\{\Delta_0(AEE), \Delta_1(AEE)\}) = 5$,
$count(\{\Delta_0(IBM), \Delta_2(AES)\}) = 5$,
$count(\{\Delta_0(IBM), \Delta_2(AEE)\}) = 6$.
At pass 3, 2 candidate 3-itemsets, $\{\Delta_0(IBM), \Delta_1(AEE), \Delta_2(AES)\}$ and $\{\Delta_0(IBM), \Delta_1(AEE), \Delta_2(AEE)\}$, are derived from L_2 according to $C_3^* = \{\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}\}$. As the subset $\{\Delta_0(AEE), \Delta_1(AES)\}$ of the first one is not frequent, we are only left with the second candidate. By scanning the database, we get its count value 5, satisfying the count requirement. The separate mining algorithm thus finishes with the result $L_3 = \{\{\Delta_0(IBM), \Delta_1(AEE), \Delta_2(AEE)\}\}$. □

Correctness. The key to the correctness of the above algorithm lies in the following lemma.

Lemma 1 *Any frequent k-itemset in L_k is included in C_k, i.e., $L_k \subseteq C_k$.*

Proof. We prove the lemma by induction.

1) When $k = 1$, $L_1 \subseteq C_1$ since C_1 includes all possible 1-itemsets in C_1^* that could be potentially frequent.
2) When $k = 2$, because the hash value $HashTable[func(x)]$ for any frequent 2-itemset $x = \{\Delta_0(i_1), \Delta_{d_2}(i_2)\}$ should not be less than $minsup * |\mathcal{T}_E|$, C_2 gives all possible normalized 2-itemsets in C_2^* that could be potentially frequent, i.e., $L_2 \subseteq C_2$.
3) Assume the lemma holds when $k = n$. Without loss of generality, assume $X = \{\Delta_{(d_1)}(i_1), \ldots, \Delta_{(d_{n-1})}(i_{n-1}), \Delta_{(d_n)}(i_n), \Delta_{(d_{n+1})}(i_{n+1})\}$ is a frequent (n+1)-itemset in L_{n+1} (i.e., $X \in L_{n+1}$). Let Y, $Y' \subset X$, where $Y = \{\Delta_{(d_1)}(i_1), \ldots, \Delta_{(d_{n-1})}(i_{n-1}), \Delta_{(d_n)}(i_n)\}$, $Y' = \{\Delta_{(d_1)}(i_1), \ldots, \Delta_{(d_{n-1})}(i_{n-1}), \Delta_{(d_{n+1})}(i_{n+1})\}$.

Input: a set of candidate itemsets $C_k^*(k = 1, 2, \ldots, RuleLen)$ identified by the *separate* mining plan; a $minsup$ support threshold.
Output: a set of frequent itemsets L_k.

k=1
1 $C_1 = \{\{\Delta_d(i)\} \mid i \in \mathcal{I} \wedge \{\Delta_d(*)\} \in C_1^*\}$;
2 set all the buckets of $HashTable$ to 0;
3 **foreach** extended transaction $\Delta_c(t) \in \mathcal{T}_E$ **do**
4 **foreach** candidate $x = \{\Delta_d(i)\} \in C_1$ **do**
5 **if** $\exists \Delta_{c+d}(t') \in \mathcal{T}_E\ (i \in t')$ **then** x.count++;
6 **foreach** $i_1 \in t$ (where $\Delta_c(t) \in \mathcal{T}_E$) **do** // build hash table
7 **foreach** $i_2 \in t''$ (where $\Delta_{c+d_2}(t'') \in \mathcal{T}_E$) **do**
8 **if** $\{\Delta_0(*), \Delta_{d_2}(*)\} \in C_2^*$ **then** $HashTable$[func($\{\Delta_0(i_1), \Delta_{d_2}(i_2)\}$)]++;
9 **endfor**
10 $L_1 = \{x : \{\Delta_d(i)\} \mid (x \in C_1) \wedge (x.count/|\mathcal{T}_E| \geq minsup)\}$;

k=2
11 $C_2 = \{x : \{\Delta_0(i_1), \Delta_{d_2}(i_2)\} \mid \{\Delta_0(*), \Delta_{d_2}(*)\} \in C_2^* \wedge \{\Delta_0(i_1)\} \in L_1 \wedge \{\Delta_{d_2}(i_2)\} \in L_1 \wedge (i_1 < i_2 \vee d_2 > 0) \wedge HashTable$[func($x$)]/$|\mathcal{T}_E| \geq minsup\}$;
12 **foreach** extended transaction $\Delta_c(t) \in \mathcal{T}_E$ **do**
13 **foreach** candidate $x = \{\Delta_0(i_1), \Delta_{d_2}(i_2)\} \in C_2$ **do**
14 **if** $(i_1 \in t) \wedge \exists \Delta_{c+d_2}(t') \in \mathcal{T}_E\ (i_2 \in t')$ **then** x.count++;
15 $L_2 = \{x : \{\Delta_0(i_1), \Delta_{d_2}(i_2)\} \mid (x \in C_2) \wedge (x.\text{count}/|\mathcal{T}_E| \geq minsup)\}$;

$3 \leq$ **k** $\leq RuleLen$
16 **for** $(k = 3;\ (L_{k-1} \neq \emptyset) \wedge (k \leq RuleLen);\ k++)$ **do**
17 C_k=E-Apriori-Gen(L_{k-1});
18 **foreach** extended transaction $\Delta_c(t) \in \mathcal{T}_E$ **do**
19 **foreach** normalized candidate $x = \{\Delta_{d_1}(i_1), \ldots, \Delta_{d_k}(i_k)\} \in C_k$ **do**
20 **if** $\exists \Delta_{c+d_j}(t') \in \mathcal{T}_E\ (i_j \in t')\ (1 \leq j \leq k)$ **then** x.count++;
21 $L_k = \{x : \{\Delta_{d_1}(i_1), \cdots, \Delta_{d_k}(i_k)\} \mid (x \in C_k) \wedge (x.\text{count}/|\mathcal{T}_E| \geq minsup)\}$;
22 **endfor**

Fig. 3. The separate mining algorithm

According to Property 2, any subset of X should be in L_n. Thus, $Y,\ Y' \in L_n$. In the join phase, the algorithm joins Y and Y' to get $X \in C_{n+1}^{join}$. Thus, after the join phase, $L_{n+1} \subseteq C_{n+1}^{join}$. By similar reasoning, the prune step , where all itemsets whose n-subsets are not in L_n are deleted from C_{n+1}^{join}, also does not discard X from C_{n+1}^{join}, leaving $X \in C_{n+1}$. Therefore, $L_{n+1} \subseteq C_{n+1}$.

Based on $1), 2)$ and $3)$, the lemma is proven. □

6.2 The joint mining algorithm

The *joint* algorithm (Fig. 4) is based on the *joint* mining plan. The differences between the two algorithms are two. First, the candidate itemsets C_k^* to be counted at each pass are different due to the different mining plans. Second, the *joint* algorithm only counts itemsets till $k = k_s$, while in the final speeding phase, it generates candidate $RuleLen$-itemsets by the *join* operation defined in the previous section (line 1), and then counts these candidates to obtain frequent $RuleLen$-itemsets (lines 2–5).

To compare with the separate mining algorithm, in the following example, we apply the joint mining algorithm to the realistic data in Table 6.

Example 16 According to the joint mining plan in Table 7, the joint mining algorithm only counts $C_1^* = \{\{\Delta_0(*)\}, \{\Delta_1(*)\}\}$ at pass 1 and $C_2^* = \{\{\Delta_0(*), \Delta_1(*)\}\}$ at pass 2. From $L_2 = \{\{\Delta_0(a), \Delta_1(d)\}, \{\Delta_0(d), \Delta_1(a)\}, \{\Delta_0(d), \Delta_1(d)\}\}$, the algorithm performs two join operations to derive 2 candidate itemsets at pass 3. That is, $\{\Delta_0(a), \Delta_1(d)\}_{\Delta_1(d)} \oplus_{\Delta_0(d)} \{\Delta_0(d), \Delta_1(a)\} = \{\Delta_0(a), \Delta_1(d), \Delta_2(a)\}$, and $\{\Delta_0(a), \Delta_1(d)\}_{\Delta_1(d)} \oplus_{\Delta_0(d)} \{\Delta_0(d), \Delta_1(d)\} = \{\Delta_0(a), \Delta_1(d), \Delta_2(d)\}$. Since only the second candidate has a count value 5, which is not less than the specified count threshold, the algorithm terminates with the frequent 3-itemset $\{\Delta_0(a), \Delta_1(d), \Delta_2(d)\}$ returned. □

Correctness. As the correctness of Fig. 3 has been demonstrated, if we can show that the target frequent $RuleLen$-itemsets being discovered by Fig. 4 is the same as those discovered by Fig. 3, the correctness of Fig. 4 can be proved accordingly.

Lemma 2 *A target frequent $RuleLen$-itemset is discovered by the joint mining algorithm if and only if it is discovered by the separate mining algorithm.*

Proof. It proceeds in two parts.

First, let $X = \{\Delta_{d_1}(x_1), \cdots, \Delta_{d_{r-1}}(x_{r-1}), \Delta_{d_r}(x_r)\}$ $(r = RuleLen)$ be a target frequent itemset discovered by the joint mining algorithm. With the downward closure property (Property 2), any of its subsets (e.g., $\{\Delta_{d_1}(x_1), \cdots, \Delta_{d_{r-2}}(x_{r-2}), \Delta_{d_{r-1}}(x_{r-1})\}$, $\{\Delta_{d_1}(x_1), \cdots, \Delta_{d_{r-2}}(x_{r-2}), \Delta_{d_r}(x_r)\}$, etc.) is also frequent. Thus, under

Input: a set of candidate itemsets $C_k^*(k = 1, 2, \ldots, k_s, RuleLen)$ identified by the *joint* mining plan; a $minsup$ support threshold.
Output: a set of frequent itemsets L_k.

$1 \leq \mathbf{k} \leq k_s$ // Identical to the *separate* algorithm

k=RuleLen // the Speeding Phase
1 C_k = Join-Gen($\{L_2, \ldots, L_{k_s}\}$);
2 **foreach** extended transaction $\Delta_c(t) \in \mathcal{T}_E$ **do**
3 **foreach** normalized candidate $x = \{\Delta_{d_1}(i_1), \ldots, \Delta_{d_k}(i_k)\} \in C_k$ **do**
4 **if** $\exists \Delta_{c+d_j}(t') \in \mathcal{T}_E\ (i_j \in t')\ (1 \leq j \leq k)$ **then** x.count++;
5 $L_k = \{x : \{\Delta_{d_1}(i_1), \cdots, \Delta_{d_k}(i_k)\} \mid (x \in C_k) \wedge (x.\text{count}/|\mathcal{T}_E| \geq minsup)\}$;

Fig. 4. The joint mining algorithm

the separate plan, such a candidate X can also be generated and counted using its candidate generation method.

Second, let $X = \{\Delta_{d_1}(x_1), \cdots, \Delta_{d_{r-1}}(x_{r-1}), \Delta_{d_r}(x_r)\}$ be a target frequent itemset discovered by the separate mining algorithm. Since X is the target $RuleLen$-itemset, which means that its contextual positions conform to the template instance, without loss of generality, assume the join operation carried out by the joint algorithm is between two itemsets, $U_{U'} \oplus_{V'} V$, with k_s= $RuleLen$-1, where $U = \{\Delta_{d_1}(u_1), \ldots, \Delta_{d_a}(u_a), \underline{\Delta_{d_{a+1}}(u_{a+1}), \ldots, \Delta_{d_b}(u_b)}\}$ $(1 \leq a < b < RuleLen)$, $V = \{\underline{\Delta_{d_{a+1}-d}(u_{a+1}), \ldots, \Delta_{d_b-d}(u_b)}, \Delta_{d_{b+1}-d}(u_{b+1}), \ldots, \Delta_{d_{u_r}-d}(u_r)\}$ $(d_{a+1} \geq d, \ldots, d_{u_r} \geq d)$, $U' = \{\Delta_{d_{a+1}}(u_{a+1}), \ldots, \Delta_{d_b}(u_b)\}$, $V' = \{\Delta_{d_{a+1}-d}(u_{a+1}), \ldots, \Delta_{d_b-d}(u_b)\}$, and the joinable distance is d.

As both algorithms share the same procedure of discovering frequent itemsets before $k = RuleLen$, to prove that X can also be detected by the joint mining algorithm, we only need to show that there exist two frequent itemsets $U \in L_{|U|}$ and $V \in L_{|V|}$ (where $1 < |U|, |V| \leq RuleLen$), on which the above join operation can be performed to derive candidate X at the final pass $k = RuleLen$.

As X is frequent, any subset of X is also frequent. Thus, we can find two frequent subsets $X_1 = \{\Delta_{d_1}(x_1), \ldots, \Delta_{d_a}(x_a), \underline{\Delta_{d_{a+1}}(x_{a+1}), \ldots, \Delta_{d_b}(x_b)}\}$ in L_b and $X_2 = \{\underline{\Delta_{d_{a+1}}(x_{a+1}), \ldots, \Delta_{d_b}(x_b)}, \Delta_{d_{b+1}}(x_{b+1}), \ldots, \Delta_{d_{u_r}}(x_r)\}$ in L_{r-a}. Since $d_{a+1} \geq d, \ \ldots, \ d_{u_r} \geq d,$ by subtracting d from each contextual position of items in X_2, and keeping the relative positions unchanged, the itemset $X_2' = \{\underline{\Delta_{d_{a+1}-d}(x_{a+1}), \ldots, \Delta_{d_b-d}(x_b)}, \Delta_{d_{b+1}-d}(x_{b+1}), \ldots, \Delta_{d_{u_r}-d}(x_r)\}$ remains frequent as well. Hereby, there exist two frequent itemsets $U = X_1, V = X_2'$, in respective frequent sets L_b and L_{r-a}, from which candidate X can be obtained and detected by the joint mining algorithm. □

7 Performance study

In this section, we report our performance study on the two different mining algorithms. As the aim of experimental evaluation in this study is to assess and gain an overall feeling about the fundamental performance of the two proposed methods, our experiments here focus on their basic behavior's investigation. Detailed comparison and analysis of specific heuristics and various choices, introduced in the paper, are left in a further study.

In the following, we first describe the methods used to generate synthetic data and template instances. Section 7.3 presents some experimental results on synthetic data. Results obtained from real data are described in Sect. 7.4.

7.1 Generation of synthetic data

The method used by this study to generate synthetic data is similar to the one used in [4] with some modifications noted below. Table 8 summarizes the parameters used and their settings.

We first generate a set L of the potentially frequent extended itemsets which may span several transactions (e.g., $\{\Delta_0(a), \Delta_1(b), \Delta_2(c)\}$), and then assign a frequent extended itemset from L to transactions. Items and their contextual positions in the first frequent itemset are chosen randomly, where item is picked up from 1 to N, and its relative contextual position is picked up from 0 to W. To model the phenomenon that frequent itemsets often have common extended items (i.e, item ID plus contextual position), some fraction of extended items in subsequent itemsets are chosen from the previous one generated. We use an exponentially distributed random variable with mean equal to the *correlation level* to decide this fraction for each itemset. The remaining extended items are picked at random. After generating all the extended items for a frequent itemset, we normalize the generated itemset by subtracting its minimal contextual point value from each contextual position in this set.

After generating the set L of potentially frequent itemsets, we then generate transactions in the database. Each transaction is assigned a series of potentially frequent itemsets. However, upon the generation of one transaction, we need to consider a list of consecutive ones starting from this transaction, as items in a frequent extended itemset may span across different transactions. For example, after selecting the frequent itemset $\{\Delta_0(a), \Delta_1(b), \Delta_2(c)\}$ for current transaction $\Delta_c(t)$, we should assign item a to t, item b to its next transaction t' which is one unit away, i.e., $\Delta_{c+1}(t')$, and item c to the transaction at the point Δ_{c+2}. If the frequent itemset picked on hand does not fit in the current or any one of its successive transactions, this itemset is put in these transactions anyway in half the cases, and the itemset enters an *unfit* queue for the next transaction the rest of the cases. Each time, we pick itemsets from this queue first according to the first-in-first-out principle. Only

Table 8. Parameters

Parameter	Meaning	Setting
	Data generation	
$\|D\|$	Number of transactions	20k - 100k
$\|T\|$	Average size of the transactions	4 - 8
$\|MT\|$	Maximum size of the transactions	8 - 12
$\|L\|$	Number of potentially frequent itemsets	2000
$\|I\|$	Average size of the potentially frequent itemsets	5 - 7
$\|MI\|$	Maximum size of the potentially frequent itemsets	8 - 10
N	Number of items	800 - 1600
W	Window size	4 - 8
	Template Generation	
$InstanceNum$	Number of template instances	3
$RuleLen$	Length of rules	4, 5
$SelfCommon\_Portion$	Portion of template instance with common relative contextual positions **within** itself	20%, 40%, 50%
$CrossCommon\_Portion$	Portion of template instances with common relative contextual positions **among** template instances	20%, 40%, 50%

when the queue is empty, do we perform random selection from the set L.

7.2 Generation of template instances

For generality, we generate various template instances using a list of parameters shown in Table 8. To model the phenomenon that some common relative contextual positions may exist within or among different template instances, we divide each template instance into three parts, containing l_1, l_2, and l_3 extended items, respectively ($l_1 + l_2 + l_3 = RuleLen$). Part I, whose contextual positions are directly selected from the previous template instance except for $\pm\Delta_b$ difference, shows the common relative positions among template instances, that is, $l_1 = \lceil CrossCommon\_Portion \times RuleLen \rceil$. For the first template instance, we generate this part randomly. Part II, whose contextual positions are selected from within the template instance's Part I except for $\pm\Delta_b$ difference, shows the common relative positions within a template instance, that is, $l_2 = \lceil SelfCommon\_Portion \times l_1 \rceil$. Contextual points in Part III are chosen randomly with $l_3 = RuleLen - l_1 - l_2$. The maximum contextual scope within each template instance is bounded by the window size W.

For example, suppose the length of the two template instances to be generated is 5 (i.e., $RuleLen$=5, $InstanceNum$=2), and the window size W=8. Let $CrossCommon\_Portion$ = 50% and $SelfCommon\_Portion$ = 30%, we have $l_1 = \lceil CrossCommon\_Portion \times RuleLen \rceil = \lceil 50\% \times 5 \rceil = 3, l_2 = \lceil SelfCommon\_Portion \times l_1 \rceil = \lceil 30\% \times 3 \rceil = 1$, and $l_3 = RuleLen - l_1 - l_2 = 5 - 3 - 1 = 1$. For the first template instance, its l_1=3 contextual positions in Part I are randomly picked up from $[0, W]$, so is the l_3=1 contextual position in Part III. Assume Part I = $\{\Delta_0, \Delta_2, \Delta_4\}$ and Part III = $\{\Delta_5\}$. The $l_2 = 1$ contextual position in Part II is selected from Part I, except for $\pm\Delta_b$ difference determined randomly. Let $\pm\Delta_b$ be $+\Delta_2$ and the selected position from Part I is Δ_4, thus Part II = $\{\Delta_6\}$. As a result, the first template instance is comprised of contextual positions $\Delta_0, \Delta_2, \Delta_4, \Delta_5, \Delta_6$. To generate the second template instance, its Part I, reflecting the common relative positions among template instances, is obtained by selecting $l_1 = 3$ contextual positions from the first template instance. Assume the selected positions are $\Delta_2, \Delta_4, \Delta_5$ and $\pm\Delta_b = -\Delta_1$, Thus, Part I = $\{\Delta_1, \Delta_3, \Delta_4\}$. Part II and Part III of the second template instance are generated in the same way as of the first template instance. Suppose Part II = $\{\Delta_4\}$ and Part III = $\{\Delta_7\}$. We have $\Delta_1, \Delta_3, \Delta_4, \Delta_4, \Delta_7$ for the second template instance, which is $\Delta_0, \Delta_2, \Delta_3, \Delta_3, \Delta_6$ after normalization.

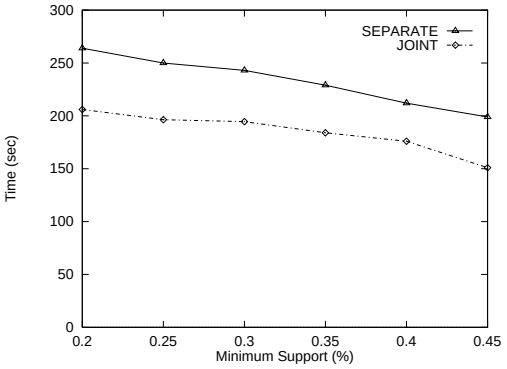
7.3 Experiments on synthetic data

Four sets of experiments were performed to investigate the performance of the *separate* algorithm and *joint* algorithm, with the emphasis on their basic behavior, scale-up properties, and some key performance influential factors, including contextual scope and template length. The machine used for the experiments is a Sun Ultra Sparc Workstation with a CPU clock rate of 164 MHz and 64 MB main memory.

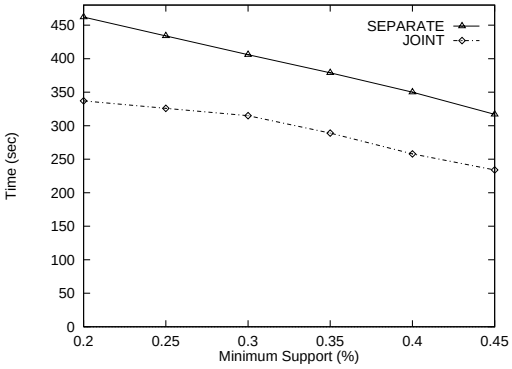
7.3.1 Basic experiments

The first set of experiments studies the basic behavior of the algorithms when the minimum support changes. Three template instances of length 4 ($InstanceNum$=3, $RuleLen$=4) are generated based on $CrossCommon/SelfCommon\_Portion$ = 50%, denoting the portion of template instances that have common relative contextual positions among/within the template instances. The speeding parameter $KLimit$ is 3 through the whole experiments.

As shown in Fig. 5, when the minimum support increases, the execution times of both algorithms decrease because of



a T5-I5-W5-N1000-D100k



b T5-I5-W7-N1000-D100k

Fig. 5. Minimum support versus execution time

Table 9. Comparison of *separate* and *joint* mining plans

separate	$\|C_1^*\| = 7$	$\|C_2^*\| = 7$	$\|C_3^*\| = 9$
mining plan	$W\text{-}Len(C_1^*) = 7$	$W\text{-}Len(C_2^*) = 7$	$W\text{-}Len(C_3^*) = 7$
joint	$\|C_1^*\| = 5$	$\|C_2^*\| = 5$	$\|C_3^*\| = 2$
mining plan	$W\text{-}Len(C_1^*) = 6$	$W\text{-}Len(C_2^*) = 6$	$W\text{-}Len(C_3^*) = 6$

the reduction in the number of candidate C_k and frequent itemsets L_k at each pass. Throughout the experiments, *joint* is always superior over *separate*. For example, in Fig. 5(b), when minimum support is 0.25%, the mining time of *separate* is around 434 s, while that of *joint* is 326 s, about 33% more time required. This is not surprising if we look at their mining plans shown in Table 9. The template instances generated for this test are "$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_1(*), \Delta_4(*)$", "$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_1(*), \Delta_2(*)$", and "$\Delta_0(*), \Delta_2(*) \Rightarrow \Delta_5(*), *\Delta_6(*)$". The joint algorithm generates its target candidates through

$$\{\Delta_0(*), \Delta_0(*), \underline{\Delta_1(*)}\}_{(3)} \oplus_{(1)} \{\underline{\Delta_0(*)}, \Delta_3(*)\} = \{\Delta_0(*), \Delta_0(*), \Delta_1(*), \Delta_4(*)\}$$

$$\{\Delta_0(*), \Delta_2(*), \underline{\Delta_5(*)}\}_{(3)} \oplus_{(1)} \{\underline{\Delta_0(*)}, \Delta_1(*)\} = \{\Delta_0(*), \Delta_2(*), \Delta_5(*), \Delta_6(*)\}$$

$$\{\Delta_0(*), \underline{\Delta_1(*)}\}_{(2)} \oplus_{(1)} \{\underline{\Delta_0(*)}, \Delta_0(*), \Delta_1(*)\} = \{\Delta_0(*), \Delta_1(*), \Delta_1(*), \Delta_2(*)\}$$

where for simplicity, i and j in $U_{(i)} \oplus_{(j)} V$ denote the ith and jth extended item in U and V, respectively.

From Table 9, we can see that at each pass, the *joint* algorithm counts less candidates which are of smaller contextual scopes than the *separate* algorithm. As a result, lots of time can be saved from searching the database. From this preliminary experiments, we note that strategies aiming at eliminating candidates, especially those with large contextual scopes, can yield significant performance benefits in mining inter-transactional associations.

7.3.2 Scale-up experiments

The second set of experiments is designed to study scale-up properties of the algorithms.

Figure 6a shows how *separate* and *joint* behave as the number of items in a database increases from 800 to 1,600. As expected, the execution times of both algorithms increase. This is because more items lead to more candidates to be counted, especially at the first two passes, resulting in more time for searching the database before finding frequent itemsets.

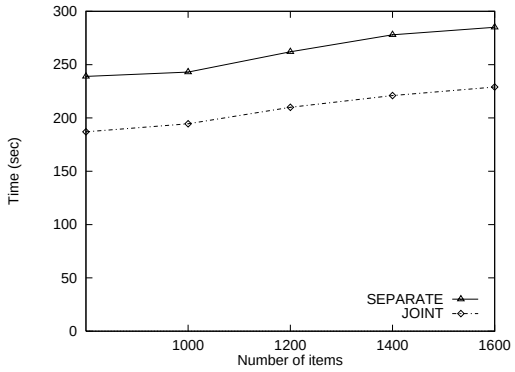
In addition, when we increase the number of transactions from 20 K to 100 K, both algorithms take longer time to scan the database and count candidate itemsets. As shown in Fig. 6(b), their execution times scale quite linearly.

Finally, we investigate the scale-up as we increase the average transaction size from 4 to 8. From the result presented in Fig. 6(c), we observe that for both algorithms, the more items per transaction, the more time needed to process. This is due to several reasons. First, given a minimum support and a set of items, when the average transaction size is large, there are more frequent 1-itemsets generated, and hence more candidate 2-itemsets need to be counted. The construction of the hash table for C_2 at pass 1 also takes longer time. Moreover, the time needed to scan every transaction of the database becomes longer, resulting in higher processing costs. For example, at average transaction size 5, the execution times of *joint* and *separate* are 194 s and 243 s, respectively, but at average transaction size 8, they increase to 338 s and 435 s, nearly 74% and 79% increment each.

7.3.3 Further experiments

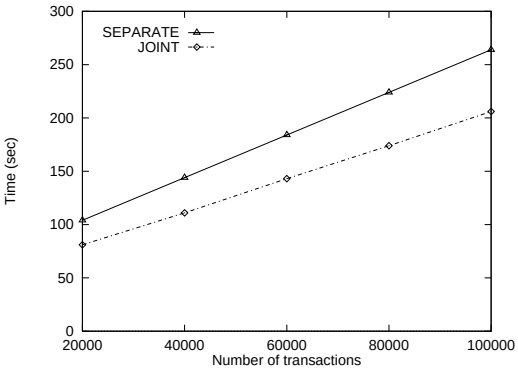
Our last experiment on synthetic database is to study the impact of contextual scope (window size) and rule length on the performance of mining algorithms. The experiment was conducted under two different templates shown in Table 10. Each template implies three template instances. The length and contextual scope of Template I are both 4, while the length and

a



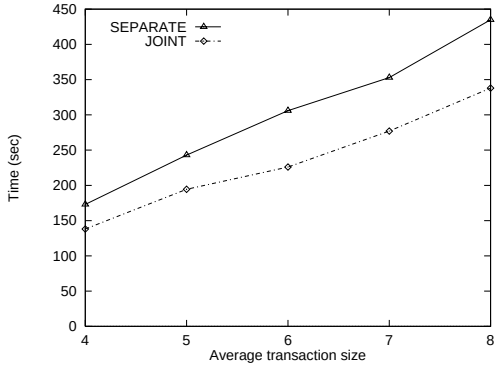
T5-I5-W5-D100k, support=0.3%

b



T5-I5-W5-N100, support=0.2%

c



I5-W5-N1000-D100k, support=0.3%

Fig. 6. Results of scale up experiments ($InstanceNum$=3, $RuleLen$=4, $CrossCommon/SelfCommon\_Portion$=50%). **a** Number of items scale-up. **b** Number of transactions scale-up. **c** Average transaction size scale-up

Table 10. Template instances for further experiment

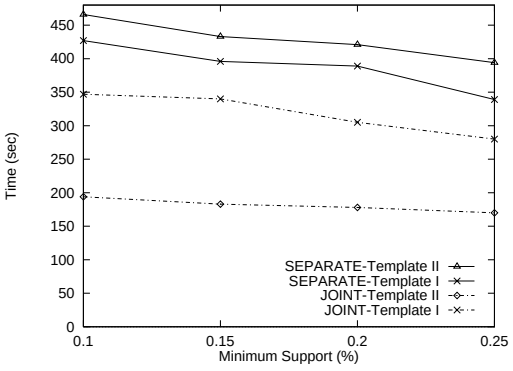
Instance	**Template I (RuleLen=4, window size=4)**	**Template II (RuleLen=5, window size=6)**
No.1	$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_3(*), \Delta_3(*)$	$\Delta_0(*), \Delta_3(*) \Rightarrow \Delta_3(*), \Delta_6(*), \Delta_6(*)$
No.2	$\Delta_0(*), \Delta_3(*) \Rightarrow \Delta_3(*), \Delta_4(*)$	$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_0(*), \Delta_2(*), \Delta_2(*)$
No.3	$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_2(*), \Delta_3(*)$	$\Delta_0(*), \Delta_2(*) \Rightarrow \Delta_4(*), \Delta_5(*), \Delta_5(*)$
separate	$\|C_1^*\| = 5, \|C_2^*\| = 5, \|C_3^*\| = 6$	$\|C_1^*\| = 7, \|C_2^*\| = 7, \|C_3^*\| = 13, \|C_4^*\| = 9$
plan	$W\text{-}Len(C_1^*) = 5$, $W\text{-}Len(C_2^*) = 5$	$W\text{-}Len(C_1^*) = 7$, $W\text{-}Len(C_2^*) = 7$
	$W\text{-}Len(C_3^*) = 5$	$W\text{-}Len(C_3^*) = 7$, $W\text{-}Len(C_4^*) = 7$
joint	$\|C_1^*\| = 4, \|C_2^*\| = 4, \|C_3^*\| = 2$	$\|C_1^*\| = 3, \|C_2^*\| = 3, \|C_3^*\| = 2$
plan	$W\text{-}Len(C_1^*) = 4$, $W\text{-}Len(C_2^*) = 4$	$W\text{-}Len(C_1^*) = 4$, $W\text{-}Len(C_2^*) = 4$
	$W\text{-}Len(C_3^*) = 4$	$W\text{-}Len(C_3^*) = 4$

contextual scope of Template II are 5 and 6, respectively. Figure 7 presents the result of the experiment.

Intuitively, the mining time under Template II should be more than that under Template I, since the itemsets indicated in Template II span more transactions and tend to be longer. In fact, the results of *separate* algorithm do justify such speculation. Table 10 shows the candidate number and the maximal contextual scope of candidates to be counted at each pass under two templates. However, the behavior of *joint* is surprisingly contrary. Looking at its search space, we find that *joint* actually counts less candidates each time by *joining* necessary itemsets, demonstrating the effectiveness of *joining* operation and *speeding* techniques in the candidate identification and generation steps. From this experiment, we note that careful selection of a search space beforehand is quite important to the inter-transactional association mining performance.

7.4 Experiments on real data

To test the applicability of inter-transactional association rules, we run the algorithms against two data sets collected from Singapore Stock Exchange (SSE). One is the WINNER set that contains the stocks whose closing prices are 3% more than the previous closing prices; and another is the LOSER set that contains other stocks. For each data set, we have 250 records



T5-I5-N1000-D100k

Fig. 7. Effects of window size and rule length (InstanceNum=3, KLimit=3)

corresponding to 250 trading days in 1996 (i.e., D=250). Although there are a few hundred stocks in SSE, we only have complete data for 84 stocks (i.e., N=84). Observing that quite a number of traders are likely to know recent stock behavior based on the information in the past few days, we use the template instance "$\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_2(*), \Delta_3(*)$" as our mining guidance.

We like to test with these two data sets, because the major trend for SSE in 1996 was down side, leading to a pretty large average transaction size (more than 70) in the LOSER data set, and a small average transaction size (less than 10) in the WINNER data set by contrast. Figures 8 and 9 show the experimental results on these two different data sets. The support threshold set for the LOSER data set is big enough (98%) as each transaction (trading day) contains lots of items (dropping stocks), while the support threshold set for the WINNER data set is small (1.5%) due to the much less items (rising stocks) per transaction (trading day).

Different from the *separate* algorithm which generates candidate $\{\Delta_0(*), \Delta_1(*), \Delta_2(*), \Delta_3(*)\}$ through $\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$ and $\{\Delta_0(*), \Delta_1(*), \Delta_3(*)\}$, *joint* algorithm directly *joins* two $\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$. Table 11 shows two different mining plans. Comparing Fig. 8 to Fig. 9, the joint mining algorithm always spends less time than the separate mining algorithm on Pass 1, 2, and 3. This is because the former needs to count much less number of candidates and the search space at each pass is smaller than the latter. At the final pass 4, the separate algorithm takes less time as it has pruned out some candidates from C_4. However, this is at the cost of spending comparatively more time on the first few passes in counting all possible subsets of 4-itemsets, leading to a higher overall mining cost.

As the LOSER set has larger average transaction size than the WINNER one, both algorithms need more time in making hash table for C_2, thus, their execution times at pass 1 are longer. From the LOSER data set, one example rule that conforms to the template is "$\Delta_0(UOL), \Delta_1(SIA) \Rightarrow \Delta_2(DBS), \Delta_3(OUT)$". It says that, if UOL goes down and SIA goes down the following day, DBS and OUT will go down on the second and third day with confidence more than 99%. Here, UOL stock represents land market, SIA stock represents loans and debentures, and both DBS and OUT represent banking market. This rule reveals the closely causal relationships among four major stocks in Singapore. As known, the land properties in Singapore play an important role in the national economic development, and therefore their decaying inevitably leads to a bad performance of loans, debentures, and banking. Such a discovered rule reveals some of the characteristics of Singapore's economic structures.

Since the WINNER data set is small, we do not have rules with large support. However, we did find some interesting rules such as
"$\Delta_0(HAISUNWT), \Delta_1(KIMENGWT) \Rightarrow \Delta_2(HAISUNWT), \Delta_3(HAISUNWT)$".
That is, if HAISUN Warrant stock and KIMENG Warrant stock (belonging to loans and bond sectors, respectively) go up on successive days, HAISUN Warrant will keep going up for the next 2 days.

Our study using stock movement data is on-going. The results obtained so far indicate that, with inter-transactional association rules, we can discover more comprehensive and interesting knowledge, and *joint* algorithm can achieve better performance than *separate* algorithm regardless of a large or small average transaction size.

8 Related work

The presented multidimensional inter-transactional association rules provide a more general view of association rule and sequential pattern mining problems [34,35,16,7,8]. First, all previous work treats data as a sequence of records in one dimension, and there are no discussions on a multidimensional context. We believe that a large number of applications exist where association is only meaningful when transaction/event is viewed along more than one dimension simultaneously, for example, in geographical applications. Second, even for one dimensional data, there are differences between the previous work and the one reported in this paper.

Traditional association rule mining. The original association rule mining proposed by Agrawal *et. al.* [1] is apparently a special case of the multi-dimensional inter-transactional association rule mining: if we omit the dimensional attributes in the transactions, and set the window size to one, the multidimensional inter-transactional association rule mining will degrade to intra-transactional association rule mining.

Sequential pattern discovery. Agrawal *et. al.* introduced the problem of mining sequential patterns from transaction databases where each record contains items bought by a particular customer, in different transactions during a period of time [5]. One sequential pattern example is "*80% of customers bought shoes* **after** *they bought shirts.*" For mining sequential patterns, transactions of each customer ordered by transaction-time are organized into one record. The problem of sequential pattern mining was further generalized to allow items to be present in a set of transactions whose transaction-times are within a user-specified time window [49], Despite this, sequential pattern mining focuses on *successive/precedent* relationships of items. On the other hand, we are interested in finding all associations across a set of transactions within all possibly **different** ranges. This part of contextual informa-

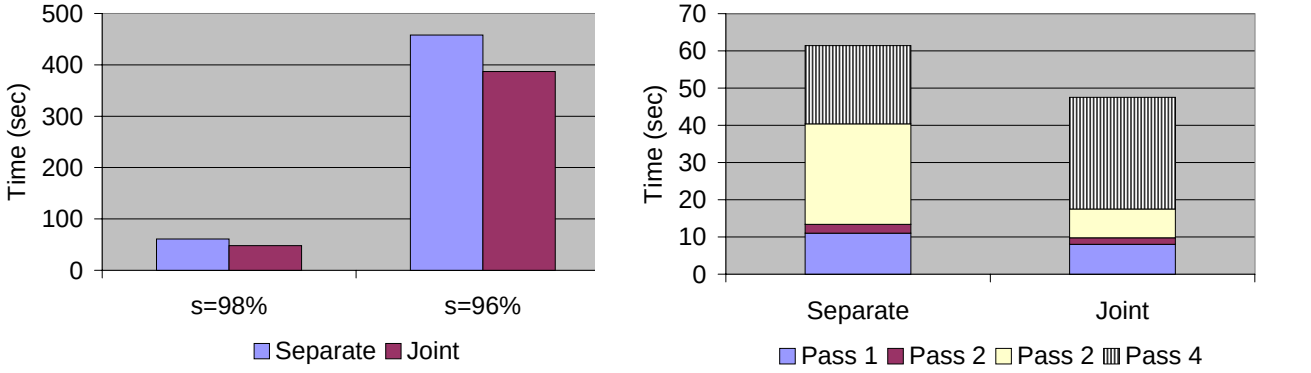


Fig. 8. Execution time on LOSER data set

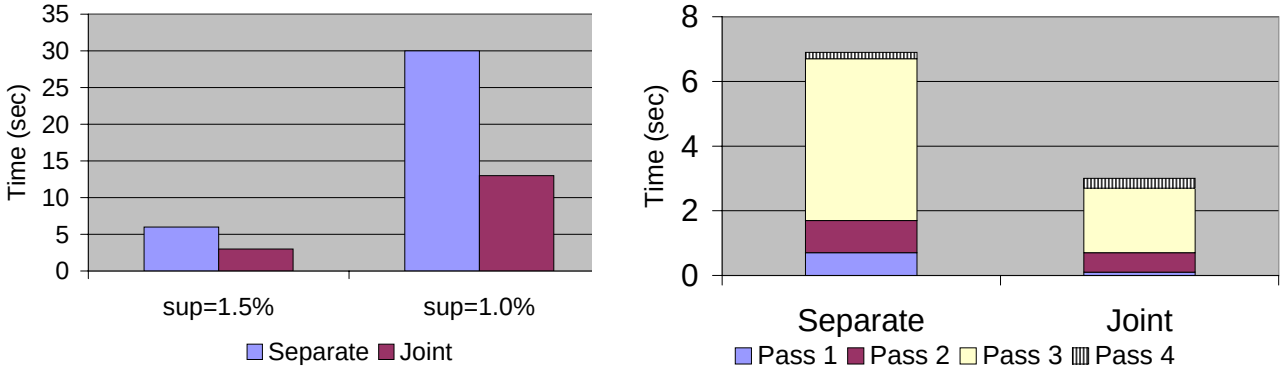


Fig. 9. Execution time on WINNER data set

Table 11. Template instances for the experiments on real-data

	Template instance: $\Delta_0(*), \Delta_1(*) \Rightarrow \Delta_2(*), \Delta_3(*)$			
separate	$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_2(*), \Delta_3(*)\}$
plan	$\{\Delta_1(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*)\}$	
	$\{\Delta_2(*)\}$	$\{\Delta_0(*), \Delta_3(*)\}$		
	$\{\Delta_3(*)\}$			
	$\lvert C_1^* \rvert = 4$	$\lvert C_2^* \rvert = 3$	$\lvert C_3^* \rvert = 2$	$\lvert C_4^* \rvert = 1$
	$W\text{-}Len(C_1^*) = 4$	$W\text{-}Len(C_2^*) = 4$	$W\text{-}Len(C_3^*) = 4$	$W\text{-}Len(C_4^*) = 4$
joint	$\{\Delta_0(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_2(*), \Delta_3(*)\}$
plan	$\{\Delta_1(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$		
	$\{\Delta_2(*)\}$			
	$\lvert C_1^* \rvert = 3$	$\lvert C_2^* \rvert = 2$	$\lvert C_3^* \rvert = 1$	$\lvert C_4^* \rvert = 1$
	$W\text{-}Len(C_1^*) = 3$	$W\text{-}Len(C_2^*) = 3$	$W\text{-}Len(C_3^*) = 3$	$W\text{-}Len(C_4^*) = 4$

tion can be explicitly captured within the inter-transactional association rule framework.

Episode rule discovery. In episode rules presented by Mannila et al. [34,35], the temporal relationship among events is expressed roughly as "*if episode P has a minimal occurrence at interval $[t, t']$ with $t' - t \leq V$, then episode Q occurs at interval $[t, t'']$ for some t'' such that $t'' - t \leq W$*", where the time bounds involved are constrained to have the same starting time t. As mentioned by the authors, only certain types of episodes with predefined predicates and time bounds are easily detected using their mining algorithms. The efficient mining of more general episode rules with arbitrary time bounds from a large sequence remains an open problem.

Rule discovery from time-series data. Das et al. proposed adaptive methods for finding rules from the preprocessed data [16]. First, a time series is converted into a discrete representation. Based on the discretized sequence, temporal patterns can be detected using previous rule-finding algorithms (such as the episode rule methods). Although this work need not define beforehand what patterns to be used, the rules they studied are rather simple and limited to the form of "*if A occurs, then B occurs within time T*". Although the

authors mentioned that the rules can be extended to a more complex form, like $A_1 \wedge \ldots \wedge A_h\ [V] \Rightarrow B\ [W]$, stating *if* A_1 *and* $\ldots$ *and* A_h *occur within* V *units of time, then* B *occurs within time* W, the rules are still limited to two windows only.

Temporal relationship mining. Compared to episode sequences and Das et al.'s rules, Bettini et al. looked for more complex event sequences from time-series data, even using different time granularities [7,8]. However, their focus is on event sequences. Neither definitions nor mining algorithms regarding the rules were discussed in the context. It is obvious that rules above certain confidence threshold can show the connections between events more clearly than event sequences alone [35].

Time-series analysis. Time-series analysis and forecasting has been an active research topic in statistics. The main purpose is to understand and model the stochastic mechanism that gives rise to an observed series, or to forecast future values of a series based on the history of that series [56,14]. DeCoste proposed a technique based on linear regression and neural network for automatic detection of anomalies in sensor data [17]. Recently, Yi *et. al* presented a fast method called MUSCLES to analyze co-evolving time sequences to enable estimation of missing/delayed/future values and outlier detection [58]. The main theme of the analysis performed in this area is different from mining rules from a large amount of data under multidimensional contexts.

Similarity retrieval from sequences. Most of sequence-related work in the database community concerns similarity search and querying, i.e., finding similar sequences that match a given pattern in some error distances, or searching all pairs of similar sequences [2,30,15,54]. Various approaches have been suggested including using the discrete Fourier transform, interpolation approximation, or defining some shape querying languages [29,**?**]. Issues such as how to detect patterns efficiently from a huge database of sequences are not the focus in this body of work.

We believe that discovering inter-transactional association rules would reveal more comprehensive and useful information which in turn can be used in broad applications for analysis, prediction, and decision making.

9 Conclusion

Mining inter-transactional association rules is a computationally intensive problem, requiring much more search effort compared to the traditional association rule mining. In order to make such association rules truly practical and extensible, in this study, we propose a template model to help users specify the interesting rules to be mined. Several optimization techniques are devised to speed up the discovery of inter-transactional association rules. Our performance study on both synthetic and real-life data sets reveal that with inter-transactional association rules, we can discover more comprehensive knowledge; and careful selection of mining plans to reduce the search space is critical to the performance of such extended association rule mining.

We plan to extend this work in several directions. Besides context constraints, we will also consider pushing item constraints into the inter-transactional association mining process. Development of efficient discovery algorithms for 2- and m-dimensional inter-transactional association rules is another future work we are working towards. It might also be interesting to study the multidimensional inter-transactional association rule mining in distributed and parallel environments.

Acknowledgements. The authors would like to thank the referees and editors-in-chief for their insightful comments which have greatly helped to improve the paper.

References

1. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 207–216, Washington D.C., USA
2. Agrawal R, Lin KI, Sawhney HS, Shim K (1995) Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In: Proc. 21st International Conference on Very Large Data Bases, pp 490–501, Zurich, Switzerland
3. Agrawal R, Shafer JC (1996) Parallel mining of association rules. IEEE Trans Knowl Data Eng 8(6):962–969
4. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proc. 20th International Conference on Very Large Data Bases, pp 478–499, Santiago, Chile
5. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proc. International Conference on Data Engineering, pp 3–14, Taipei, Taiwan
6. Baralis E, Psaila G (1997) Designing templates for mining association rules. J Intell Inf Syst 9(1):7–32
7. Bettini C, Wang X, Jajodia S (1996) Testing complex temporal relationships involving multiple granularities and its applications to data mining. In: Proc. 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp 68–78, Montreal, Canada
8. Bettini C, Wang X, Jajodia S (1998) Mining temporal relationships with multiple granularities in time sequences. Data Eng 21(1):32–38
9. Brin S, Motwani R, Silverstein C (1997) Beyond market basket: generalizing association rules to correlations. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 265–276, Tucson, Ariz., USA
10. Chakrabarti S, Sarawagi S, Dom B (1998) Mining surprising patterns using temporal description length. In: Proc. 24th International Conference on Very Large Data Bases, pp 606–617, New York, USA
11. Cheung D, Han J, Ng V, Wong CY (1996) Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proc. International Conference on Data Engineering, pp 106–114, New Orleans, La., USA
12. Cheung DW, Ng VT, Fu AW, Fu YJ (1996) Efficient mining of association rules in distributed databases. IEEE Trans Knowl Data Eng 8(6):911–922
13. Cohen E, Datar M, Fujiwara S, Gionis A, Indyk P, Motwani R, Ullman JD, Yang C (2000) Finding interesting associations without support pruning. In: Proc. International Conference Data Engineering, pp 489–499, Calif., USA
14. Cryer JD (1985) Time series analysis. Duxbury, MBelmont, Calif., USA
15. Das G, Gunopulos D, Mannila H (1997) Finding similar time series. In: Proc. International Conference on Principles of Data Mining and Knowledge Discovery, pp 88–100, Norway

16. Das G, Lin KI, Mannila H, Renganathan G, Smyth P (1998) Rule discovery from time series. In: Proc. 2nd International Conference on Knowledge Discovery and Data Mining, pp 16–22, New York, USA
17. DeCoste D (1997) Mining multivariate time-series sensor data to discover behavior envelopes. In: Proc. 3rd International Conference on Knowledge Discovery and Data Mining, pp 151–154, Newport Beach, Calif., USA
18. Dehaspe L, Toivonen H (1999) Discovery of frequent DATALOG patterns. Data Min Knowl Discovery 3(1):7–36
19. Fang M, Shivakumar N, Garcia-Molina H, Motwani R, Ullman JD (1998) Computing iceberg queries efficiently. In: Proc. 24th International Conference on Very Large Data Bases, pp 299–310, New York, USA
20. Fukuda T, Morimoto Y, Morishita S, Tokuyama T (1996) Data mining using two-dimensional optimized association rules: schema, algorithms, and visualization. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 13–23, Montreal, Canada
21. Fukuda T, Morimoto Y, Morishita S, Tokuyama T (1996) Mining optimized association rules for numeric attributes. In: Proc. 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp 182–191, Montreal, Canada
22. Han EH, Karypis G, Kumar V (1997) Scalable parallel data mining for association rules. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 277–288, Tucson, Ariz., USA
23. Han J, Fu Y (1995) Discovery of multiple-level association rules from large databases. In: Proc. 21st International Conference on Very Large Data Bases, pp 420–431, Zurich, Switzerland
24. Han J, Fu Y (1995) Meta-rule-guided mining of association rules in relational databases. In: Proc. 1st International Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases, pp 39–46, Singapore
25. Kamber M, Han J, Chiang JY (1997) Metarule-guided mining of multi-dimensional association rules using data cubes. In: Proc. International Conference on Knowledge Discovery and Data Mining, pp 207–210, Calif., USA
26. Klemettinen M, Mannila H, Ronkainen P, Toivonen H, Verkamo AI (1994) Finding interesting rules from large sets of discovered association rules. In: Proc. 3rd International Conference on Information and Knowledge Management, pp 401–408, Gaithersburg, Md., USA
27. Lakshmanan LVS, Ng R, Han J, Pang A (1999) Optimization of constrained frequent set queries with 2-variable constraints. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 157–168, USA
28. Lent B, Swami A, Widom J (1997) Clustering association rules. In: Proc. International Conference on Data Engineering, pp 220–231, Birmingham, UK
29. Li CS, Yu PS, Castelli V (1996) HierachyScan: a hierachical similarity search algorithm for databases of long sequences. In: Proc. International Conference on Data Engineering, pp 546–553, New Orleans, La., USA
30. Lin L, Risch T (1998) Querying continuous time sequences. In: Proc. 24th International Conference on Very Large Data Bases, pp 170–181, New York, USA
31. Liu B, Hsu W, Ma Y (1999) Mining association rules with multiple minimum supports. In: Proc. ACM SIGKDD International Conference Knowledge Discovery and Dara Mining, pp 125–134, Calif., USA
32. Lu H, Feng L, Han J (2000) Beyond intra-transactional association analysis: mining multi-dimensional inter-transaction association rules. ACM Trans Inf Syst 18(4):423–454
33. Lu H, Han J, Feng L (1998) Stock movement prediction and n-dimensional inter-transaction association rules. In: Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pp 12:1–12:7, Seattle, Wash., USA
34. Mannila H, Toivonen H (1996) Discovering generalized episodes using minimal occurrences. In: Proc. 2nd International Conference on Knowledge Discovery and Data Mining, pp 146–151, Portland, Ore., USA
35. Mannila H, Toivonen H, Verkamo A (1997) Discovering frequent episodes in event sequences. Data Min Knowl Discovery 3(1):259–289
36. Meo R, Psaila G, Ceri S (1996) A new SQL-like operator for mining association rules. In: Proc. 22th International Conference on Very Large Data Bases, pp 122–133, Mumbai, India
37. Miller RJ, Yang Y (1997) Association rules over interval data. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 452–461, Tucson, Ariz., USA
38. Ng R, Lakshmanan LVS, Han J, Pang A (1998) Exploratory mining and pruning optimizations of constrained association rules. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 13–24, Seattle, Wash., USA
39. Özden B, Ramaswamy A, Silberschatz A (1998) Cyclic association rules. In: Proc. International Conference on Data Engineering, pp 412–421, Fla., USA
40. Park JS, Chen MS, Yu PS (1995) An effective hash-based algorithm for mining association rules. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 175–186, San Jose, Calif., USA
41. Park JS, Chen MS, Yu PS (1995) Mining association rules with adjustable accuracy. Technical Report IBM Research Report
42. Park JS, Chen MS, Yu PS (1996) Data mining for path traversal patterns in a web environment. In: Proc. 16th Conference on Distributed Computing Systems, pp 385–392, Hong Kong
43. Ramaswamy S, Mahajan S, Silberschatz A (1998) On the discovery of interesting patterns in association rules. In: Proc. 24th International Conference on Very Large Data Bases, pp 368–379, New York, USA
44. Rastogi R, Shim K (1998) Mining optimized association rules with categorical and numerical attributes. In: Proc. International Conference on Data Engineering, pp 503–512, Florida, USA
45. Savasere A, Omiecinski E, Navathe S (1995) An efficient algorithm for mining association rules in large databases. In: Proc. 21st International Conference on Very Large Data Bases, pp 432–443, Zurich, Switzerland
46. Silverstein C, Brin S, Motwani R, Ullman JD (1998) Scalable techniques for mining causal structures. In: Proc. 24th International Conference on Very Large Data Bases, pp 594–605, New York, USA
47. Silverstein C, Brin S, Motwani R, Ullman JD (2000) Scalable techniques for mining causal structures. Data Min Knowl Discovery 4(2/3):163–192
48. Srikant R, Agrawal R (1995) Mining generalized association rules. In: Proc. 21st International Conference on Very Large Data Bases, pp 409–419, Zurich, Switzerland
49. Srikant R, Agrawal R (1996) Mining quantitative association rules in large relational tables. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 1–12, Montreal, Canada
50. Srikant R, Vu Q, Agrawal R (1997) Mining association rules with item constraints. In: Proc. 3rd International Conference on Knowledge Discovery and Data Mining, pp 67–73, Newport Beach, Calif., USA
51. Toivonen H (1996) Sampling large databases for association rules. In: Proc. 22th Conference on Very Large Data Bases, pp 134–145, Mumbai, India

52. Tsur D, Ullman JD, Abitboul S, Clifton C, Motwani R, Nestorov S (1998) Query flocks: a generalization of association-rule mining. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 1–12, Seattle, Wash., USA
53. Tung KH, Lu H, Han J, Feng L (1999) Breaking the barrier of transactions: mining inter-transaction association rules. In: Proc. ACM SIGKDD International Conference Knowledge Discovery and Data Mining, pp 297–301
54. Wang JTL, Chirn GW, Marr TG, Shapiro B, Shasha D, Zhang K (1994) Combinatorial pattern discovery for scientific data: some preliminary results. In: Proc. ACM SIGMOD International Conference on Management of Data, pp 115–125, Minn., USA
55. Wang K, He Y, Han J (2000) Mining frequent itemsets using support constraints. In: Proc. 26st International Conference Very Large Data Bases, pp 43–52, Cairo, Egypt
56. Weigend A, Gerschenfeld N (1994) Time series prediction: forecasting the future and understanding the past. Addison-Wesley, Reading, Mass., USA
57. Yi BK, Jagadish HV, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. In: Proc. 16th International Conference on Data Engineering, pp 201–208, Fla., USA
58. Yi BK, Sidiropoulos ND, Johnson T, Jagadish HV, Faloutsos C, Biliris A (2000) Online data mining for co-evolving time sequences. In: Proc. 16th International Conference on Data Engineering, pp 13–22, Calif., USA
59. Zaki MJ, Parthasarathy S, Ogihara M, Li W (1997) New algorithms for fast discovery of association rules. In: Proc. 3rd International Conference on Knowledge Discovery, Data Mining, pp 283–286, Newport Beach, Calif., USA