Mining Inter-Transaction Associations with Templates *

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

[1]Hong Kong Polytechnic University, China. cslfeng@comp.polyu.edu.hk
[2]Hong Kong University of Science and Technology, China. luhj@cs.ust.hk
[3]Australian National University, Australia. yu@cs.anu.edu.au
[4]Simon Fraser University, Canada. han@cs.sfu.ca

Abstract

Multi-dimensional, inter-transaction association rules extend the traditional association rules to describe more general associations among items with multiple properties cross 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-transaction association rules tends to be extremely large, mining inter-transaction associations poses more challenges on efficient processing than mining intra-transaction associations. In order to make such association mining truly practical and computationally tractable, in this study, we present a template model to help users declare the interesting inter-transaction associations *to be mined. With the guidance of templates, several optimization techniques are devised to speed up the discovery of inter-transaction association rules. We show, through a series of experiments, that these optimization techniques can yield significant performance benefits.*

1 Introduction

Since the problem of mining association rules was introduced in [AIS93], a large amount of work has been done in various directions including efficient, Apriori-like mining methods, mining generalized, multi-level, or quantitative association rules, association rule mining query languages, constraint-based rule mining, incremental maintenance of discovered association rules, parallel and distributed mining, mining correlations and casual structures, association rule mining, etc. Despite these efforts, there is an important form of association rules which are useful but could not be discovered with the existing association rule mining framework.

Taking stock market as an example, we can construct a share price movement database as follows: each trading day has one record in the database and each counter corresponds to a field in the record. The value of a field tells whether the price of the counter goes up or down. Applying the association rules as mentioned above, we can discover the rules like

R_1 : *When the prices of IBM and SUN go up, the price of Microsoft will most likely go up on the same day.*

While association rule such as R_1 reflects some relationship among the prices, its role in price prediction is limited; and traders may be more interested in the following type of rules:

R_2 : *If the prices of IBM and SUN go up, Microsoft's will most likely go up* **the next day.**

Unfortunately, current association rule miners cannot discover this type of rules. This is because of the fundamental difference between the rules like R_2 and those such as R_1, which we will refer to as classical association rules. The classical association rules express the associations among items within the *same transaction*, such as items purchased by a customer or share price movement within a day. On the other hand, rule R_2 represents some associations *among the field values from different transaction records*. To distinguish these two types of associations, we name the classical associations as **intra-transaction associations** and the latter as **inter-transaction associations**.

In [LHF98], we introduced multi-dimensional inter-transaction association rules mining, and discussed its related properties in comparison with [BWJ96, BWJ98, DLM+98, MT96, MTV97]. A preliminary performance study was conducted by entension of Apriori [AS96]. From the initial results, we found that multi-dimensional

*The first author's work is jointly supported by Hong Kong Polytechnic University, Departmental Research Grant 351/849 GS 781 and Hong Kong Government University Grants Council - CERG PolyU 6074/98E. The second author's work is partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (No DAG97/98.EG33) and a grant from the National 973 project of China (No. G1998030414)

CIKM '99 11/99 Kansas City, MO, USA

inter-transaction association rules do bring us more comprehensive knowledge than traditional intra-transaction association rules, but this is at the expense of higher computational cost. In order to make inter-transaction association rule mining truly practical and computationally tractable, we extend this piece of work in this study and propose a template model to help such rule discovery. Previous work on traditional association rules demonstrated the effectiveness of constraint/query-based association mining [NLHP98, SVA97, TUA+98, MPC96, BP97]. It is applicable to inter-transaction associations as well. Moreover, users may also have certain interesting inter-transaction intervals 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.

Thus, one contribution of this paper is to provide users with a set of constructors to specify the interesting *inter-transaction associations*, so that mining can be focused and the cost incurred is proportionate to what the user wants and gets. Another contribution of the paper is that we develop several optimization techniques, i.e., *joining, converging* and *speeding*, for mining inter-transaction 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.

2 Multi-Dimensional Inter-Transaction Association Rules

In this section, we provide some notations and background information for multi-dimensional inter-transaction association rules.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_s\}$ denote a set of literals, called items, on an m-dimensional attribute schema. A transaction database $\mathcal{T}$ is a set of transactions $\{t_1, t_2, \cdots, t_n\}$ where t_i is a subset of $\mathcal{I}$. On the other hand, an m-dimensional space is defined as a finite subset of N^m where N is the set of nonnegative integers. A mapping function *dim* maps an m-dimensional attribute item onto a point in the m-dimensional space. It is important to know that the m-dimensional space provides a unified platform onto which any m-dimensional attribute values, such as colors, can be mapped through a user-defined mapping function. In the following discussion, for simplicity, we only discuss inter-transaction association rule mining on the m-dimensional space.

Let n_i and n_j be two points in the m-dimensional space, a binary relation, $n_i \prec n_j$, is given which implies n_i precedes n_j in the lexicographical order.[1] In addition, a relative distance between n_i and n_j is denoted as $\Delta(n_i, n_j)$.[2] In this paper, we also use $\Delta(n_i)$ or simply Δ_i for $\Delta(n_0, n_i)$ where n_0 is a reference point in the m-dimensional space. (Note that n_i, $\Delta(n_i)$ and Δ_i can be used interchangeably with respect to a reference point.) We call a m-dimensional attribute item i_k at the point Δ_j in the m-dimensional space an **extended item** and denote it as $\Delta_j(i_k)$. In general, two extended items $\Delta_i(i_k)$ and $\Delta_j(i_k)$ are not equal if $\Delta_i \neq \Delta_j$. We call a transaction t_i at the point Δ_j in the m-dimensional space an **extended transaction** and denote it as $\Delta_j(t_i)$. The set of all possible extended-items, $\mathcal{I}_e$, is defined as a set of $\Delta_j(i_k)$ for $i_k \in \mathcal{I}$ at all points Δ_j in the m-dimensional space. $\mathcal{T}_e$ is the set of all extended transactions in the m-dimensional space.

A reference point of a subset of $\mathcal{I}_e$ is the smallest reference point for all the extended items in the subset.[3] Normalization is a process to reposition the extended items in either a set of extended items or a set of extended transactions regarding to the reference point of the set in question. A set of extended transactions is called a normalized extended transaction set if the included extended items are normalized.

Table 1: A 1-dimensional extended transaction database

Trans.	Date	Items	Extended Trans.	Extended Items
t_1	day_1	a, b, c	$\Delta_0(t_1)$	$\Delta_0(a), \Delta_0(b), \Delta_0(c)$
t_2	day_2	c, d, e	$\Delta_1(t_2)$	$\Delta_1(c), \Delta_1(d), \Delta_1(e)$
t_3	day_3	a, b	$\Delta_2(t_3)$	$\Delta_2(a), \Delta_2(b)$
t_4	day_4	a, b, c, e	$\Delta_3(t_4)$	$\Delta_3(a), \Delta_3(b), \Delta_3(c), \Delta_3(e)$
t_5	day_5	b, c, d, e	$\Delta_4(t_5)$	$\Delta_4(b), \Delta_4(c), \Delta_4(d), \Delta_4(e)$

Example 2.1 *A simple traditional database is transformed into a 1-dimensional extended transaction database as shown in Table 1, through a mapping function which maps day_i onto a point $i-1$ in the 1-dimensional space. Let $T'_e = \{\Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4)\}$. Following the definition, the reference point of T'_e is 1. The extended items in T'_e are normalized to this reference point. We say T'_e contains a set of normalized extended items:* $\{\Delta_0(c), \Delta_0(d), \Delta_0(e), \Delta_1(a), \Delta_1(b), \Delta_2(a), \Delta_2(b), \Delta_2(c), \Delta_2(e)\}$.

Definition 2.1 A multi-dimensional inter-transaction association rule *is an implication of the form $X \Rightarrow Y$, where $X \subset \mathcal{I}_e$, $Y \subset \mathcal{I}_e$, and $X \cap Y = \emptyset$.*

[1] The binary relation needs to be provided in order to specify queries. The properties need to be precisely specified. However, the properties of this binary relation does not affect our mining mechanism.

[2] The relative distance is defined as follows. Let $n_i = \langle v_1, v_2, \cdots, v_m \rangle$ and $n_j = \langle u_1, u_2, \cdots, u_m \rangle$. $\Delta(n_i, n_j) = \langle u_1 - v_1, u_2 - v_2, \cdots, u_m - v_m \rangle$.

[3] Let $n_i = \langle v_1, v_2, \cdots, v_m \rangle$ and $n_j = \langle u_1, u_2, \cdots, u_m \rangle$. The smallest reference point of the two points is $\langle \min(u_1, v_1), \min(u_2, v_2), \cdots, \min(u_m, v_m) \rangle$.

Based on Definition 2.1, a rule that predicts the stock price movement, "*if the price of stock 'a' increases one day, and the price of stock 'c' increases the following day, then most probably the price of stock 'e' will increase on the fourth day*" can be expressed by a 1-dimensional inter-transaction association rule as: $\Delta_0(a), \Delta_1(c) \Rightarrow \Delta_3(e)$.

Table 2: Three normalized extended transaction sets that contain $\Delta_0(a)$ and $\Delta_1(c)$. Note: RF refers to reference point.

A set of extended {transactions T'_e	The RF of T'_e	Normalized extended items in T'_e
$\{\Delta_0(t_1), \Delta_1(t_2)\}$	0	$\{\Delta_0(a), \Delta_0(b), \Delta_0(c)$ $\Delta_1(c), \Delta_1(d), \Delta_1(e)\}$
$\{\Delta_2(t_3), \Delta_3(t_4)\}$	2	$\{\Delta_0(a), \Delta_0(b)$ $\Delta_1(a), \Delta_1(b), \Delta_1(c), \Delta_1(e)\}$
$\{\Delta_3(t_4), \Delta_4(t_5)\}$	3	$\{\Delta_0(a), \Delta_0(b), \Delta_0(c), \Delta_0(e)$ $\Delta_1(b), \Delta_1(c), \Delta_1(d), \Delta_1(e)\}$

Similar to intra-transaction association rules, we use *support* and *confidence* as two major measurements. Traditionally, the support of a rule $X \Rightarrow Y$ is the fraction of transactions that contain $X \cup Y$ over the whole transactions, and the confidence of the rule is the fraction of transactions containing X that also contain Y. Let's consider the rule "$\Delta_0(a), \Delta_1(c) \Rightarrow \Delta_3(e)$" applied to Example 2.1. Table 2 shows three normalized extended transaction sets that contain $\{\Delta_0(a), \Delta_1(c)\}$. The only normalized extended transaction set that includes $\{\Delta_0(a), \Delta_1(c), \Delta_3(e)\}$ is $\{\Delta_0(t_1), \Delta_1(t_2), \Delta_2(t_3), \Delta_3(t_4)\}$. Therefore, the support and confidence are 1/5 and 1/3, respectively, where 1 is the number of normalized extended transaction sets that include $\{\Delta_0(a), \Delta_1(c), \Delta_3(e)\}$, 5 is the total number of transactions, and 3 is the number of normalized extended transaction sets that contain $\{\Delta_0(a), \Delta_1(c)\}$.

Definition 2.2 *Given two subsets of $\mathcal{I}_e$, X and Y. Let T_{xy} be a set of normalized extended transaction sets that contain $X \cup Y$, T_x be a set of normalized extended transaction sets that contain X, and $\mathcal{T}_e$ be the set of all extended transactions in the database. The* **support** *and* **confidence** *of an inter-transaction association rule $X \Rightarrow Y$ is defined as follows: $support(X \Rightarrow Y) = |T_{xy}|/|\mathcal{T}_e|$ and $confidence(X \Rightarrow Y) = |T_{xy}|/|T_x|$.*

Figure 1 depicts a 2-dimensional extended transaction database. There are totally four items, a, b, c, and d. For simplicity, we denote a point in the 2-dimensional space using $\Delta_{x,y}$ with respect to a certain reference point.[4] The database contains extended transactions such as $\Delta_{0,0}(t_1) = \{\Delta_{0,0}(a), \Delta_{0,0}(b), \Delta_{0,0}(c)\}$, $\Delta_{1,0}(t_2) = \{\Delta_{1,0}(b)\}$, ..., $\Delta_{4,3}(t_{20}) = \{\Delta_{4,3}(a)\}$. Upon such an

[4]Recall Δ_i is used for a point n_i in m-dimensional space regarding a reference point. The (x, y) in $\Delta_{x,y}$ are used to index a point in 2-dimensional space.

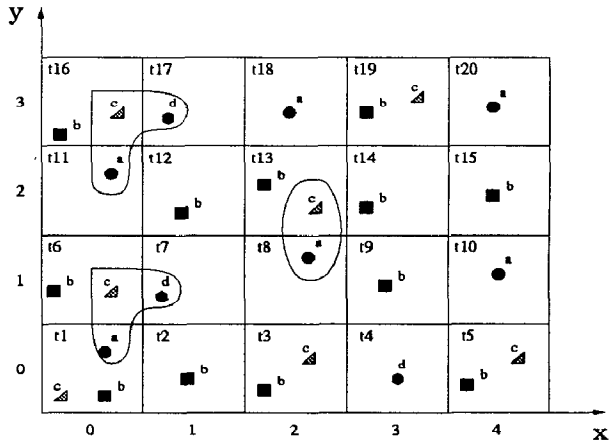


Figure 1: Graphical representation of a 2-dimensional transaction database.

extended transaction database, the support and confidence of the rule "$\Delta_{0,0}(a), \Delta_{0,1}(c) \Rightarrow \Delta_{1,1}(d)$" are 2/20 and 2/3, respectively, since there are totally 3 normalized extended transaction sets: $\{\Delta_{0,0}(t_1), \Delta_{0,1}(t_6), \Delta_{1,1}(t_7)\}$, $\{\Delta_{0,2}(t_{11}), \Delta_{0,3}(t_{16}), \Delta_{1,3}(t_{17})\}$, and $\{\Delta_{2,1}(t_8), \Delta_{2,2}(t_{13}), \Delta_{3,2}(t_{14})\}$, containing $\{\Delta_{0,0}(a), \Delta_{0,1}(c)\}$, and 2 of them contain $\{\Delta_{0,0}(a), \Delta_{0,1}(c), \Delta_{1,1}(d)\}$.

3 Template Model for Inter-Transaction Association Rules

A frequently encountered problem in association rule mining is that mining systems may return quite a large number of rules. With inter-transaction associations, which capture more knowledge than traditional 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 *inter-transaction association rules* are to be mined. In the following, we will first define some operators used in the inter-transaction association rule templates.

First, consider two points Δ_i and Δ_j in the m-dimensional space. Assume that the relative distance function and the binary relation $\prec$ are given. The following boolean comparison operators can be defined.

- $equal(\Delta_i, \Delta_j)$ is true iff the relative distance between the two is zero.
- $precedence(\Delta_i, \Delta_j)$ is true iff $\Delta_i \prec \Delta_j$.
- $adjacent(\Delta_i, \Delta_j)$ is true iff $precedence(\Delta_i, \Delta_j)$ is true and the relative distance between the two is 1.

Second, let $w_i = \langle \Delta_s, \Delta_e \rangle$ be a template window in the m-dimensional space. Assume $size(w_i)$ is a function that returns the size of the template window w_i. Some binary operators can be defined on the template

windows including *intersect, union* and *difference*. Let $w_i = \langle \Delta_{s_i}, \Delta_{e_i} \rangle$ and $w_j = \langle \Delta_{s_j}, \Delta_{e_j} \rangle$. For example, $intersect(w_i, w_j)$ will return a template window $w_k = \langle \max(\Delta_{s_i}, \Delta_{s_j}), \min(\Delta_{e_i}, \Delta_{e_j}) \rangle$. Also we assume that boolean binary operators such as *overlap* and *inclusion* between two template windows can be defined. Furthermore, given a Δ_i and a window w_i, a member function $member(\Delta_i, w_i)$ will return true if Δ_i is included in the template window.

A **template** is an expression $\nabla_1(P_1) \wedge \cdots \wedge \nabla_m(P_m) \Rightarrow \nabla_{m+1}(P_{m+1}) \wedge \ldots \wedge \nabla_{m+n}(P_{m+n})$ $(\mathcal{B}_e, \mathcal{B}_I)$ where $\nabla_i(P_i)$ is an **extended item variable**, $\mathcal{B}_I$ is a set of constraints upon items, and $\mathcal{B}_e$ is a set of constraints upon the m-dimensional space. Note that all ∇_i refer to the same Δ_0 as the reference point. The above rule template states that "find those association rules that satisfy both $\mathcal{B}_e$ and $\mathcal{B}_I$". As item constraints have been extensively studied in [NLHP98, SVA97, TUA+98, MPC96, BP97, HF95], we will focus on $\mathcal{B}_e$ in this paper. Without loss of generality, the boolean expression $\mathcal{B}_e$ is in the *conjunctive normal form*, i.e., $D_1 \wedge D_2 \wedge \cdots \wedge D_m$, where each D_i is of the form $\alpha_{i_1} \vee \alpha_{i_2} \vee \cdots \vee \alpha_{i_n}$. Let ∇_i and ∇_j be two extended item variables, w_i and w_j be two template windows, and v be an integer, each allowed α_{i_j} is drawn from the following classes of constraints:

- Constant Constraints: a) $\nabla_i = \langle v_1, v_2, \cdots, v_n \rangle$ where v_i is a nonnegative integer in the i-th dimension; b) $w_i = \langle \nabla_s, \nabla_e \rangle$ where both ∇_s and ∇_e are points.
- Constraints between points: $equal(\nabla_i, \nabla_j)$, $precedence(\nabla_i, \nabla_j)$ and $adjacent(\nabla_i, \nabla_j)$.
- Constraints between template windows: $overlap(w_i, w_j)$ and $inclusion(w_i, w_j)$. (A template window can be obtained from other windows using *intersect, union* and *difference* operators.)
- Constraints between a point and a template window: $member(\nabla_i, w_i)$.
- Constraints involving an integer: $size(w_i)\ \theta\ v$ and $d(\nabla_i, \nabla_j)\ \theta\ v$, where θ is one of the boolean operators, $=, \neq, <, \leq, >, \geq$. (Aggregate functions *min* and *max* can be prefixed to $size(w_i)$ and $d(\nabla_i, \nabla_j)$ as well.)

Example 3.1 Suppose users show interest in those rules like "*When stock 'a' rises, and within the following 2 days another different stock also rises, then which stock will most probably rise one week later following the second rise*", we can describe such constraint using the template $\nabla_1(a) \wedge \nabla_2(P_2) \Rightarrow \nabla_3(P_3)$ $(\mathcal{B}_e, \mathcal{B}_I)$, where $\mathcal{B}_I$: $P_2 \neq A$ and $\mathcal{B}_e$: $\nabla_1 = \Delta_0 \wedge w_1 = \langle \Delta_1, \Delta_2 \rangle \wedge member(\nabla_2, w_1) \wedge d(\nabla_2, \nabla_3) = 7$.

Apparently, rules like "$\Delta_0(a), \Delta_1(P_2) \Rightarrow \Delta_8(P_3)$" and "$\Delta_0(a), \Delta_2(P_2) \Rightarrow \Delta_9(P_3)$" satisfy the above template. We call each such rule format as a **template instance**. A template may imply several conforming template instances.

Example 3.2 The template $\nabla_1(P_1) \wedge \nabla_2(P_2) \Rightarrow \nabla_3(P_3)$ $(\mathcal{B}_e, \mathcal{B}_I)$, where $\mathcal{B}_I$: $\emptyset$ and $\mathcal{B}_e$: $\nabla_1 = \Delta_0 \wedge equal(\nabla_2, \nabla_1) \wedge (d(\nabla_3, \nabla_1) = 7 \vee d(\nabla_3, \nabla_1) = 14)$, expresses such interest as "*When two items appear on the same day, a certain item will happen one week or two week later.*". The template instances implied by this template include "$\Delta_0(P_1), \Delta_0(P_2) \Rightarrow \Delta_7(P_3)$" and "$\Delta_0(P_1), \Delta_0(P_2) \Rightarrow \Delta_{14}(P_3)$".

4 Mining Inter-Transaction Association Rules Under Templates

In this section, we give an overview of mining inter-transaction association rules. The mining process can be divided into four phases: *template interpretation, mining planning, large-itemset discovery*, and *association rule generation*.

Phase-1 (Template Interpretation): As mentioned before, users describe the interesting inter-transaction associations through templates, and each such template implies one or several *template instances*, either of which the rules discovered later must conform to. For example, the rule template given in Example 3.2 represents two template instances: "$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_7(*)$" and "$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_{14}(*)$". Since its $\mathcal{B}_I$ is empty, we use * as a wide card. Before starting mining process, we need first interpret and translate template, which is conveyed by a boolean expression, into template instance(s) as above so as to provide guidance for mining algorithms.

Phase-2 (Mining Planning): Like traditional association rule mining, we first discover large extended itemsets, and then derive rules from these large itemsets. Different from traditional itemsets where all items are within the same transaction, a k-itemset under the circumstance of inter-transaction associations may span several transactions, resulting in a much larger search space for the discovery of large itemsets than ever. For example, to get rules "$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_7(*)$" and "$\Delta_0(*), \Delta_0(*) \Rightarrow \Delta_{14}(*)$", we need identify large 3-itemsets by counting candidate 3-itemsets $C_3^* = \{\{\Delta_0(*), \Delta_0(*), \Delta_7(*)\}, \{\Delta_0(*), \Delta_0(*), \Delta_{14}(*)\}\}$ across every 8 and 15 consecutive transactions.[5] Also, candidates C_3^* will be generated by joining lower-level large itemsets L_2^*, and so on. The purpose of this phase is to identify least amount of candidate itemsets 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 Section 5.

[5] In the paper, we use C_k^* to represent candidate itemsets with relative addresses in the m-dimensional space, and C_k to represent candidate itemsets with both relative addresses and specific items. The same for L_k^* and L_k.

Phase-3 (Large-Itemset Discovery): In this phase, we find the set of all large extended itemsets identified in Phase-2. Two algorithms for generating large itemsets based on different mining plans are proposed.

Phase-4 (Association Rule Generation): Using the large itemsets, we can find the desired inter-transaction association rules. The generation of inter-transaction association rules is similar to the generation of classical association rules [AS94].

5 Mining Planning Phase

Mining inter-transaction association rules is a computationally intensive problem, requiring considerable search efforts compared to the classical association rule mining. Because of this, careful selection of mining plan is important to the overall performance and further the practicability of inter-transaction association rules. In this section, we first describe a straight-forward mining plan under the guidance of template instances. Several optimizations are then explored in order to reduce search complexity for such association mining.

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^* to be counted at each pass k ($1 \leq k \leq RuleLen$) for each individual template instance, in a similar way as Apriori-Gen does [AS94]. That is, candidate $C_k^* = \{\Delta_{u_1}(*), \Delta_{u_2}(*), \cdots, \Delta_{u_{k-1}}(*), \Delta_{u_k}(*)\}$ is generated by joining two large k-1-itemsets in L_{k-1}^*, i.e., $\{\Delta_{u_1}(*), \Delta_{u_2}(*), \cdots, \Delta_{u_{k-2}}(*), \Delta_{u_{k-1}}(*)\}$ and $\{\Delta_{u_1}(*), \Delta_{u_2}(*), \cdots, \Delta_{u_{k-2}}(*), \Delta_{u_k}(*)\}$, where the first $k-2$ items of both have the same distances relative to a reference point. Due to the nice property that "*any subset of a large itemset must be large*", L_{k-1}^* also includes those k-1-itemsets $\{\Delta_{u_2}(*), \Delta_{u_3}(*), \cdots, \Delta_{u_{k-1}}(*), \Delta_{u_k}(*)\}$, $\{\Delta_{u_1}(*), \Delta_{u_3}(*), \cdots, \Delta_{u_{k-1}}(*), \Delta_{u_k}(*)\}$, $\cdots$, $\{\Delta_{u_1}(*), \cdots, \Delta_{u_{k-3}}(*), \Delta_{u_{k-1}}(*), \Delta_{u_k}(*)\}$ 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 5.1 *Suppose we have two template instances after template translation: "*$\Delta_0(P_1), \Delta_2(P_2) \Rightarrow \Delta_4(P_3), \Delta_6(P_4)$*" and "*$\Delta_0(P_1), \Delta_1(P_2) \Rightarrow \Delta_3(P_3), \Delta_5(P_4)$*". Table 3 illustrates all candidate itemsets identified by the* separate *method. To derive rules conforming to the first template instance, we need calculate the support of candidate 4-itemsets* $\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$ *in* C_4^* *to get* L_4^*. *Before that, two candidate 3-itemsets:* $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ *and* $\{\Delta_0(*), \Delta_2(*), \Delta_6(*)\}$, *shall be counted in order to generate candidate 4-itemsets. For pruning purpose,* $\{\Delta_0(*), \Delta_4(*), \Delta_6(*)\}$ *(the subset of* $\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$*) is also included in* C_3^*. *(Note that, another subset* $\{\Delta_2(*), \Delta_4(*), \Delta_6(*)\}$ *is actually equal to* $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$*). Candidate itemsets under the second template instance are decided in a similar way.*

5.2 Optimizations: Joining, Converging and Speeding

As counting each candidate itemset in Table 3 requires searching several transactions, the mining cost is extremely high. Here, we ask one question: can we just count those least amount of *necessary* candidate itemsets? In this subsection, we discuss various optimization techniques, *joining, converging* and *speeding*, to tackle such issue. These techniques are based on the following two facts. First, reducing the number of candidate extended itemsets can substantially reduce the running time of large itemsets detection. Second, reducing the size of the window that covers the candidate extended itemsets in the m-dimensional space can substantially reduce the running time of large itemsets detection.

In inter-transaction associations, apart from items, their relative addresses are also captured within an itemset. For example, from two itemsets $U = \{\Delta_0(*), \Delta_2(*), \Delta_3(*)\}$ and $V = \{\Delta_0(*), \Delta_1(*), \Delta_7(*)\}$, we know that the last two items of U appear in two consecutive transactions, similar to the first two items in V. Based on such common relative positions (although one is (Δ_2, Δ_3) and the other is (Δ_0, Δ_1)), We can consider joining them in order to reduce the overall cost. In the following, we define n-**joinable** condition and give a **join** operator.

Definition 5.1 *Given two itemsets:* $U = \{\Delta_{u_1}(u_1), \Delta_{u_2}(u_2), \cdots, \Delta_{u_s}(u_s)\}$ *and* $V = \{\Delta_{v_1}(v_1), \Delta_{v_2}(v_2), \cdots, \Delta_{v_t}(v_t)\}$. *Let* $U' \in 2^U$ *and* $V' \in 2^V$. *V and U are n-***joinable** *iff there exist U' and V' that satisfy the following conditions: (a)* $|U'| = |V'|$, *and (b) for a given non-negative integer* Δ_d, *there exists a one-to-one mapping between U' and V' such that for any* $\Delta_{u_i}(u_i) \in U'$ *there is a* $\Delta_{v_i}(v_i) \in V'$ *where* $v_i = u_i$ *and* $\Delta_{u_i} = \Delta_{v_i} + \Delta_d$. *We call* Δ_d *the* **joinable distance**, *and n the* **joinable size**.

Example 5.2 *For two itemsets* $V = \{\Delta_0(a), \Delta_2(b), \Delta_4(c), \Delta_6(d)\}$ *and* $U = \{\Delta_0(b), \Delta_4(d), \Delta_5(e)\}$, *there exist such* $V' = \{\Delta_2(b), \Delta_6(d)\}$ *and* $U' = \{\Delta_0(b), \Delta_4(d)\}$. *Therefore, V and U are 2-joinable. The joinable distance is* Δ_2.

Definition 5.2 *Let* $U = \{\Delta_{u_1}(u_1), \Delta_{u_2}(u_2), \cdots, \Delta_{u_s}(u_s)\}$ *and* $V = \{\Delta_{v_1}(v_1), \Delta_{v_2}(v_2), \cdots, \Delta_{v_t}(v_t)\}$ *be n-joinable on* $U' \subseteq U$ *and* $V' \subseteq V$. *U* **join** *V is given as*

Table 3: The *separate* mining plan: candidate itemsets to be counted. Note mW is the minimal window that covers C_i^* in the m-dimensional space, and the *size* gives the size of the window.

$\Delta_0(P_1), \Delta_2(P_2) \Rightarrow \Delta_4(P_3), \Delta_6(P_4)$			
$\{\Delta_0(*)\}$ $\{\Delta_2(*)\}$ $\{\Delta_4(*)\}$ $\{\Delta_6(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$ $\{\Delta_0(*), \Delta_4(*)\}$ $\{\Delta_0(*), \Delta_6(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ $\{\Delta_0(*), \Delta 2(*), \Delta_6(*)\}$ $\{\Delta_0(*), \Delta_4(*), \Delta_6(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$
$\Delta_0(P_1), \Delta_1(P_2) \Rightarrow \Delta_3(P_3), \Delta_5(P_4)$			
$\{\Delta_0(*)\}$ $\{\Delta_1(*)\}$ $\{\Delta_3(*)\}$ $\{\Delta_5(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$ $\{\Delta_0(*), \Delta_2(*)\}$ $\{\Delta_0(*), \Delta_3(*)\}$ $\{\Delta_0(*), \Delta_4(*)\}$ $\{\Delta_0(*), \Delta_5(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*)\}$ $\{\Delta_0(*), \Delta_1(*), \Delta_5(*)\}$ $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ $\{\Delta_0(*), \Delta_3(*), \Delta_5(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*), \Delta_5(*)\}$
C_1^*	C_2^*	C_3^*	C_4^*
$\lvert C_1^* \rvert = 7$	$\lvert C_2^* \rvert = 6$	$\lvert C_3^* \rvert = 6$	$\lvert C_4^* \rvert = 2$
$size(mW(C_1^*)) = 7$	$size(mW(C_2^*)) = 7$	$size(mW(C_3^*)) = 7$	$size(mW(C_4^*)) = 7$

$U\ _{U'} \oplus_{V'}\ V = \{\Delta_{w_1}(w_1), \Delta_{w_2}(w_2), \cdots, \Delta_{w_m}(w_m)\}$. *Let* $m = s + t - n$, *and* Δ_d *be the joinable distance.*

- $\Delta_{w_i}(w_i) = \Delta_{u_i}(u_i)$ *for* $1 \leq i \leq s$.
- $\Delta_{w_i}(w_i) = \Delta_{v_j}(v_j)$ *for* $s < i \leq m$, *where* $\Delta_{w_i} = \Delta_{v_j} + \Delta_d$ *for* $\Delta_{v_j}(v_j) \in (V - V')$.

Example 5.3 *Let* $U = \{\Delta_0(a), \Delta_2(b), \Delta_4(c)\}$ *and* $V = \{\Delta_0(b), \Delta_2(c), \Delta_4(d)\}$. *The result of* U *join* V *on* U' *and* V', *for* $U' = \{\Delta_2(b), \Delta_4(c)\}$ *and* $V' = \{\Delta_0(b), \Delta_2(c)\}$, *is* $\{\Delta_0(a), \Delta_2(b), \Delta_4(c), \Delta_6(d)\}$. *Let* $U_1 = \{\Delta_0(a), \Delta_1(b)\}$ *and* $V_1 = V$. *The result of* U_1 *join* V_1 *on* $U_1' = \{\Delta_1(b)\}$ *and* $V_1' = \{\Delta_0(b)\}$ *is* $\{\Delta_0(a), \Delta_1(b), \Delta_3(c), \Delta_5(d)\}$.

Compared to the *separate* mining plan which generates candidate $\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$ using $\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$ and $\{\Delta_0(*), \Delta_2(*), \Delta_6(*)\}$, the *joining* method is apparently superior for the following two reasons. First, it needs less number of candidate itemsets. Second, the sizes of minimal windows are much smaller. Along with the *joining* operation, the next question arises: which itemsets are suitable to be *joined* in order to generate candidate *RuleLen*-itemsets? One technique we use is called *converging* which converges all template instances together. To illustrate, let's see two templates "$\Delta_0(P_1), \Delta_2(P_2) \Rightarrow \Delta_4(P_3), \Delta_6(P_4)$" and "$\Delta_0(P_1), \Delta_1(P_2) \Rightarrow \Delta_3(P_3), \Delta_5(P_4)$" given in Example 5.1. We note that some common relative addresses exist either within or among template instance(s). 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 address information (i.e, every iterative transactions). 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 4 shows the *joining* mining plan, which derives the target C_4^* with joins as given in Example 5.3. For $k < 4$, the way to generate C_k^* is similar to that in the *separate* planning. Comparing Table 3 with Table 4, the later exhibits a much smaller search space in terms of both candidate numbers and the minimal windows that cover C_k^* at each iteration, indicating that converging different template instances and selecting appropriate candidate joining plans can reduce the mining cost significantly.

The other optimization technique we use is called **speeding** which aims at limiting the number of database scan 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 joins. First, let $U_1 = \{\Delta_0(*), \Delta_1(*)\}$ and $V_1 = U_1$. Then, $U_1\ _{U_1'} \oplus_{V_1'} V_1 = \{\Delta_0(*), \Delta_1(*), \Delta_2(*)\}$ for $U_1' = \{\Delta_1(*)\}$ and $V_1' = \{\Delta_0(*)\}$. Second, let U_2 be the result of $U_1\ _{U_1'} \oplus_{V_1'} V_1$, and $V_2 = \{\Delta_0(*), \Delta_1(*), \Delta_1(*)\}$. Then, $U_2\ _{U_2'} \oplus_{V_2'} V_2 = \{\Delta_0(*), \Delta_1(*), \Delta_1(*), \Delta_3(*), \Delta_3(*)\}$ for $U_2' = \{\Delta_2(*)\}$ and $V_2' = \{\Delta_0(*)\}$. After the 3^{rd} scan, we can directly construct C_5^* without L_4^*.

5.3 Joint Mining Plan

The *joining*, *converging* and *speeding* techniques lead us to the *joint* mining plan. As the *separate* mining plan follows the same way as Apriori-Gen to generate target candidates, its correctness has been extensively proven in [AS94]. Here, we show that the target large itemsets being discovered under the *joint* plan will be the same as those discovered under the corresponding *separate* plan.

Theorem 5.1 *A target large itemset is discovered under the joint mining plan iff it is discovered under the separate mining plan.*

Based on two different mining plans mentioned before, two large-itemset discovery algorithms, namely *separate* and *joint*, are proposed accordingly.

Table 4: The *joining* mining plan: candidate itemsets to be counted. Note mW is the minimal window that covers C_i^*, and the *size* gives the size of the window.

$\Delta_0(P_1), \Delta_2(P_2) \Rightarrow \Delta_4(P_3), \Delta_6(P_4)$			
$\{\Delta_0(*)\}$ $\{\Delta_2(*)\}$ $\{\Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*)\}$ $\{\Delta_0(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*), \Delta_6(*)\}$
$\Delta_0(P_1), \Delta_1(P_2) \Rightarrow \Delta_3(P_3), \Delta_5(P_4)$			
$\{\Delta_0(*)\}$ $\{\Delta_1(*)\}$ $\{\Delta_2(*)\}$ $\{\Delta_4(*)\}$	$\{\Delta_0(*), \Delta_1(*)\}$ $\{\Delta_0(*), \Delta_2(*)\}$ $\{\Delta_0(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_2(*), \Delta_4(*)\}$	$\{\Delta_0(*), \Delta_1(*), \Delta_3(*), \Delta_5(*)\}$
C_1^*	C_2^*	C_3^*	C_4^*
$\lvert C_1^* \rvert = 4$	$\lvert C_2^* \rvert = 3$	$\lvert C_3^* \rvert = 1$	$\lvert C_4^* \rvert = 2$
$size(mW(C_1^*)) = 5$	$size(mW(C_2^*)) = 5$	$size(mW(C_3^*)) = 5$	$size(mW(C_4^*)) = 7$

Table 5: Parameters

Parameter	Meaning	Setting
Data Generation		
$\lvert D \rvert$	number of transactions	20k - 100k
$\lvert T \rvert$	average size of the transactions	4 - 8
$\lvert MT \rvert$	maximum size of the transactions	8 - 12
$\lvert L \rvert$	number of potentially large itemsets	2000
$\lvert I \rvert$	average size of the potentially large itemsets	5 - 7
$\lvert MI \rvert$	maximum size of the potentially large itemsets	8 - 10
N	number of items	800 - 1600
W	maximum interval scope (window size)	4 - 8
Template Generation		
InstanceNum	number of template instances	3
RuleLen	length of rules	4, 5
X_Overlap	portion of overlapped addresses within rule	20%, 40%, 50%
Y_Overlap	portion of overlapped addresses among rules	20%, 40%, 50%

6 Performance Study

To assess the performance of the proposed mining algorithms, we conducted several experimental studies.

6.1 Generation of Synthetic Data

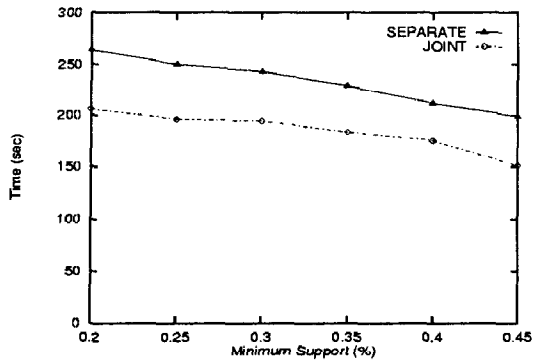
The method used by this study to generate synthetic data is similar to the one used in [AS94] with some modifications noted below. Table 5 summarizes the parameters used and their settings.

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

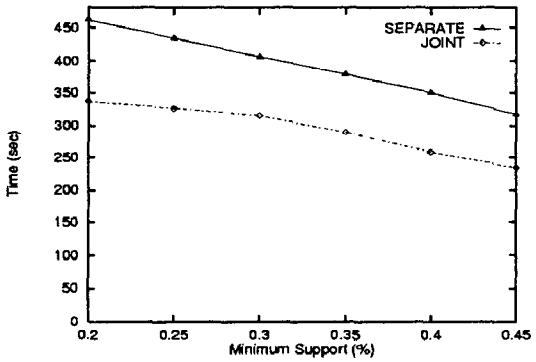
After generating the set L of potentially large itemsets, we then generate transactions in the database. Each transaction is assigned a series of potentially large itemsets. However, upon the generation of one transaction, we need consider a list of consecutive ones starting from this transaction, as items in a large itemset may span across different transactions. For example, after selecting the large itemset $\{\Delta_0(a), \Delta_1(b), \Delta_2(c)\}$ for current transaction T_c, we should assign item a to T_c, item b to its next transaction T_{c+1}, and item c to T_{c+2}. If the large 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 when the queue is empty, do we perform random selection from the set L.

6.2 Generation of Template Instances

For generality, we generate various template instances using a list of parameters shown in Table 5. To model the phenomenon that some common relative addresses may exist within or among different template instances, we divide each template instance into three parts: l_1, l_2, and l_3 ($l_1 + l_2 + l_3 = RuleLen$). Part I, whose addresses are directly selected from previous template instance except for $\pm\Delta_b$ difference, shows the common addresses among template instances. Let $l_1 = \lceil Y\_Overlap \times RuleLen \rceil$. For the first template instance, we generate this part randomly. Part II shows the common relative addresses within a template. $l_2 = \lceil X\_Overlap \times l_1 \rceil$. Addresses in Part III are chosen randomly with $l_3 = RuleLen - l_1 - l_2$. The maximum interval scope within



(a) T5-I5-W5-N1000-D100k



(b) T5-I5-W7-N1000-D100k

Figure 2: Minimum support versus execution time

each template instance is limited by the window size W.

6.3 Experiments on Synthetic Data

Four sets of experiments were performed to investigate the performance of *separate* algorithm and *joint* algorithm. 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.

6.3.1 Basic Experiment

The first set of experiments studies the basic behavior of the algorithms when the minimum support changes. 3 template instances of length 4 ($InstanceNum = 3$, $RuleLen = 4$) are generated based on $X/Y\_Overlap = 50\%$. The speeding parameter $KLimit$ is 3 through the whole experiments.

As shown in Figure 2, when the minimum support increases, the execution times of both *separate* and *joint* decrease because of reduction in the number of candidate C_k and large itemsets L_k at each pass. Throughout the experiments, *joint* is always superior over *separate*. For example, in Figure 2(b), when minimum support is 0.25%, the mining time of *separate* is around 434 seconds, while that of *joint* is 326 seconds, about 33% more time required. This is not surprising if we look at their mining plans shown in Table 6. 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

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

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

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

where for simplicity, i and j in ${}_{(i)}\oplus_{(j)}$ are the i-th and j-th extended item in U and V, respectively.

From Table 6, we can see that at each pass, the *joint* algorithm can count less candidates which are of smaller interval scopes than the *separate* algorithm. As a result, lots of time can be saved from searching database. From this preliminary experiments, we note that strategies aiming at eliminating unnecessary candidates, especially those with large interval scopes, can yield significant performance benefits in mining inter-transaction associations.

Table 6: Comparison of *separate* and *joint* mining plans

mining plan	separate	joint
$\lvert C_1^* \rvert$	7	5
$\lvert C_2^* \rvert$	7	5
$\lvert C_3^* \rvert$	9	2
$size(mW(C_1^*))$	7	6
$size(mW(C_2^*))$	7	6
$size(mW(C_3^*))$	7	6

6.3.2 Further Experiment

Also, we study the impact of maximum interval scope (window size) and rule length on the performance of mining algorithms. The experiment was conducted under two different templates shown in Table 7. Each template implies three template instances. The length and interval scope of Template I are both 4, while the length and interval scope of Template II are 5 and 6, respectively. Figure 6 presents the results 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 7 shows the candidate number and the maximal interval 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

Table 7: Template instances for further experiment

Instance	Template I (RuleLen=4, Interval=4)	Template II (RuleLen=5, Interval=6)
No.1	$\Delta_0(*) \wedge \Delta_0(*) \Rightarrow \Delta_3(*) \wedge \Delta_3(*)$	$\Delta_0(*) \wedge \Delta_3(*) \Rightarrow \Delta_3(*) \wedge \Delta_6(*) \wedge \Delta_6(*)$
No.2	$\Delta_0(*) \wedge \Delta_3(*) \Rightarrow \Delta_3(*) \wedge \Delta_4(*)$	$\Delta_0(*) \wedge \Delta_0(*) \Rightarrow \Delta_0(*) \wedge \Delta_2(*) \wedge \Delta_2(*)$
No.3	$\Delta_0(*) \wedge \Delta_1(*) \Rightarrow \Delta_2(*) \wedge \Delta_3(*)$	$\Delta_0(*) \wedge \Delta_2(*) \Rightarrow \Delta_4(*) \wedge \Delta_5(*) \wedge \Delta_5(*)$
separate plan	$\lvert C_1^* \rvert = 5, \lvert C_2^* \rvert = 5, \lvert C_3^* \rvert = 6$ $size(mW(C_1^*)) = 5$, $size(mW(C_2^*)) = 5$ $size(mW(C_3^*)) = 5$	$\lvert C_1^* \rvert = 7, \lvert C_2^* \rvert = 7, \lvert C_3^* \rvert = 13, \lvert C_4^* \rvert = 9$ $size(mW(C_1^*)) = 7$, $size(mW(C_2^*) = 7$ $size(mW(C_3^*)) = 7$, $size(mW(C_4^*)) = 7$
joint plan	$\lvert C_1^* \rvert = 4, \lvert C_2^* \rvert = 4, \lvert C_3^* \rvert = 2$ $size(mW(C_1^*)) = 4$, $size(mW(C_2^*)) = 4$ $size(mW(C_3^*)) = 4$	$\lvert C_1^* \rvert = 3, \lvert C_2^* \rvert = 3, \lvert C_3^* \rvert = 2$ $size(mW(C_1^*)) = 4$, $size(mW(C_2^*) = 4$ $size(mW(C_3^*)) = 4$

counts less candidates each time by *joining* necessary itemsets, demonstrating the effectiveness of *joining* operation and *speeding* techniques in candidate identification and generation. From this experiment, we note that careful selection of search space beforehand is quite important to the inter-transaction association mining performance.

7 Conclusion

In this paper, we present a more general form of association rules, *multi-dimensional inter-transaction association rules*. These rules can represent not only the associations of items *within* transactions, but also associations of items *among* different transactions. Mining inter-transaction association rules is a computationally intensive problem, requiring much more search efforts 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-transaction association rules. Our performance study reveals that with inter-transaction association rules, we can discover more comprehensive knowledge; and careful selection of search space beforehand is critical to the effectiveness and efficiency of such association mining.

References

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 207–216, Washington D.C., USA, May 1993.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Intl. Conf. Very Large Data Bases*, pages 478–499, Santiago, Chile, September 1994.

[AS96] R. Agrawal and J.C. Shafer. Parallel mining of association rules: Design, implementation and experience. Technical Report IBM Research Report RJ 10004, 1996.

[BP97] E. Baralis and G. Psaila. Designing templates for mining association rules. *Journal of Intelligent Information Systems*, 9(1):7–32, 1997.

[BWJ96] C. Bettini, X. Wang, and S. Jajodia. 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*, pages 68–78, Montreal, Canada, June 1996.

[BWJ98] C. Bettini, X. Wang, and S. Jajodia. Mining temporal relationships with multiple granularities in time sequences. *Data Engineering*, 21(1):32–38, March 1998.

[DLM+98] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proc. of the second International Conference on Knowledge Discovery and Data Mining*, pages 16–22, New York, USA, August 1998.

[HF95] J. Han and Y. Fu. Meta-rule-guided mining of association rules in relational databases. In *Proc. 1st Intl. Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases*, pages 39–46, Singapore, December 1995.

[LHF98] H. Lu, J. Han, and L. Feng. Stock movement prediction and n-dimensional inter-transaction association rules. In *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 12:1–12:7, Seattle, Washington, June 1998.

[MPC96] R. Meo, G. Psaila, and S. Ceri. A new sql-like operator for mining association rules. In *Proc. 22th Intl. Conf. Very Large Data Bases*, pages 122–133, Mumbai, India, September 1996.

[MT96] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proc. 2nd Intl. Conf. Knowledge Discovery and Data Mining*, pages 146–151, Portland, Oregon, August 1996.

[MTV97] H. Mannila, H. Toivonen, and A. Verkamo. Discovering frequent episodes in event sequences. Technical Report Department of Computer Science, Report C-1997-15, February 1997.

[NLHP98] R. Ng, L.V.S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 13–24, Seattle, Washington, June 1998.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proc. 3rd Intl. Conf. Knowledge Discovery and Data Mining*, pages 67–73, Newport Beach, California, August 1997.

[TUA+98] D. Tsur, J.D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: a generalization of association-rule mining. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 1–12, Seattle, Washington, June 1998.