Under consideration for publication in Knowledge and Information Systems

WAR: Weighted Association Rules for Item Intensities

Wei Wang[1], Jiong Yang2, and Philip Yu[3]

1Computer Science Department, UNC, Chapel Hill, NC, USA;
2Computer Science Department, UIUC, Urbana, IL, USA;
3IBM T.J. Watson Research Center, Hawthorne, NY, USA

Abstract. In this paper, we extend the traditional association rule problem by allowing a weight to be associated with each item in a transaction to reflect the interest/intensity of each item within the transaction. In turn, this provides us with an opportunity to associate a weight parameter with each item in a resulting association rule, we call them weighted association rules (WAR). One example of such a rule might be 80% of the people buying more than 3 bottles of soda will also be likely to buy more than 4 packages of snack food, while a conventional association rule might just be 60% of people buying soda will be also likely to buy snack food. Thus WARs can not only improve the confidence of the rules, but also provide a mechanism to do more effective target marketing by identifying or segmenting customers based on their potential degree of loyalty or volume of purchases. Our approach mines WARs by first ignoring the weight and finding the frequent itemsets (via a traditional frequent itemset discovery algorithm), and followed by introducing the weight during the rule generation. Specifically, the rule generation is achieved by partitioning the weight domain space of each frequent itemset into fine grids, and then identifying the popular regions within the domain space to derive WARs. This approach does not only support the batch mode mining, i.e., finding WARs for the dataset, but also supports the interactive mode, i.e., finding and refining WARs for a given (set) of frequent itemset(s).

Keywords: Weighted association rules; numeric attributes; density; ordered shrinkage

1. Introduction

Data mining is the extraction of implicit knowledge and discovery of interesting characteristics and patterns that are not explicitly presented in the data. These

Received xxx
Revised xxx
Accepted xxx

techniques can play an important role in presenting data in a concise manner and accommodating data semantics. During recent years, one active topic in this field is association rule discovery (Agrawal et al, 1993) (Agrawal and Srikant, 1994) (Savasere et al, 1995) (Brin et al, 1997a) (Brin et al, 1997b) (Mannila and Toivonen, 1997) (Mannila et al, 1997) (Bayardo, 1998). In general, the problem can be modeled as follows: let $\Im$ be a set of items and $\Re$ be a set of transactions, each of which consists of a subset of items in $\Im$. An association rule is an implication in the form of $X \longrightarrow Y$ where X and Y are sets of disjoint itemsets. We say that, $X \longrightarrow Y$ holds in $\Re$ with *support* s and *confidence* c if $s\%$ transactions contain all items in $X \cup Y$ and $c\%$ transactions which contain all items in X also contain all items in Y. The goal of association rule mining process is to find all these rules with respect to some minimum s and c.

These conventional association rules have been widely used in many application domains. The following are two examples.

1. *Market basket.* Each merchandise in a supermarket can be viewed as an item, and the set of merchandises that a customer purchases at one time period can be viewed as a transaction. The association rules represent the likelihood of merchandises being purchased together by a customer. Thus, this knowledge can be used to guide sale promotions. For example, if there is a high confidence between *soda* and *snacks*, then the supermarket may reduce the price of soda to increase the sale of snacks.
2. *Web trace.* Each web page can be viewed as an item, and the set of web pages accessed by a user within a short period of time can be treated as a transaction. The association rules may reveal the likelihood of web pages being accessing together by a user. This information can be utilized by web caching structure (Yang et al, 2000) and web page recommendation system (Balabanovic and Shoham, 1997).

However, the traditional association rules focus on binary attribute. In other words, this approach only considers whether an item is present in a transaction, but does not take into account the weight/intensity of an item within a transaction. For example, a customer may purchase 10 bottles of soda and 5 bags of snacks and another may purchase 4 bottles of soda and 1 bag of snacks at a time. However, these two transactions will be treated the same in the conventional association rule approach. This could lead to loss of some vital information, i.e., intensity or weight of each item in a transaction. For example, if a customer buys more than 7 bottles of soda, he is likely to purchase 3 or more bags of snacks. Otherwise, the purchase tendency of soda is not strong. The traditional association rule can not express this type of relationship. With this knowledge, the supermarket manager may set a promotion such as "if a customer buys 8 bottles of soda, he can get two free bags of snacks."

In this paper, we first extend the tradition association rule problem by allowing a weight to be associated with each item in a transaction to reflect interest/intensity of each item within the transaction. In turn, this provides us with an opportunity to associate a weight parameter with each item in a resulting association rule, we call them weighted association rule (WAR). For example, $soda[4,6] \longrightarrow snacks[3,5]$ is a weighted association rule indicating that if a customer purchases soda in the quantity between 4 and 6 bottles, he is likely to purchase 3 to 5 bags of snacks. Thus WAR can not only improve the confidence of the rules, but also provide a mechanism to do more effective target marketing by identifying or segmenting customers based on their potential degree of loyalty or volume of purchases.

In addition, WAR can also be used in many other applications. For example, in a web environment, a WAR $NFL[10, 20] \longrightarrow NBA[15, 30]$ may represent that if a user has accessed NFL web pages between 10 and 20 times, then he is likely to access the web pages of NBA between 15 and 30 times in the same period of time. This suggests that it may be worthwhile to prefetch and/or cache the NBA web page after a client visits the NFL web page 10 to 20 times. This knowledge can also be useful to recommend potential interesting web sites to qualified clients.

Previous work dealing with numerical attributes includes the quantitative association rule approach (Srikant and Agrawal, 1996a) (Lent et al, 1997) (Aumann and Lindell, 1999) and optimized association rule approach (Fukuda et al, 1996a) (Fukuda et al, 1996b) (Rastogi and Shim, 1998) (Rastogi and Shim, 1999) (Brin et al, 1999). These approaches are not designed for the weighted association rules. For example, quantitative association rule approach is designed for mining a large relational database with quantitative attributes in each record. In the problem we study in this paper, there can be a very large number of items and every item has a numerical attribute, although only a small fraction of items are present in a transaction. Thus, in our approach, the frequent itemsets are first generated (without considering weights).

The next step is to find the weighted association rules for each frequent itemset. Our goal is to segment the weight domain of each item in the itemset so that rules with higher confidence can be discovered. For an itemset consisting of i items, every transaction which supports this itemset can be mapped into a point of i dimensional space. Each dimension corresponds to the weight domain of an item in the itemset. If there are ω distinct values on each weight domain, then there are $O(\omega^2)$ distinct weight intervals associated with each item. As a result, there could be $O(\omega^{2\times i})$ possible weighted association rules that may be derived from a given i-itemset. It is thus infeasible to validate all of them. Note that each weight interval combination essentially corresponds to a hyper box in the domain space. Unfortunately, even though the support metric can provide efficient pruning in computing frequent itemset, it is not effective in pruning the search space of weighted association rules for a given frequent itemset. This is due to the fact that the support of a hyper box is always greater than that of any of its sub-boxes. Note that the confidence metric can not provide pruning effect since it does not hold the downward closure property. Thus, some additional mechanism is necessary to enable an efficient mining algorithm.

On the other hand, the specified weight interval of each attribute in a weighted association rule should also coincide with the natural distribution of the data and human intuition. In most case, only the weight interval combinations that represent a significant number of transactions are interesting. Our goal can be transformed to find highly populated regions. Therefore, we use another metric *density* for both purposes. We will explain later in detail that density is used not only in *geometric pruning* to reduce the search space, but also in discovering the regions which are heavily populated by transactions.

As a result, the weight domain space of each frequent itemset is partitioned into fine grids. A density threshold is used to separate the transaction concentrating regions from the rest. WARs can be identified based on these “dense” regions. In reality, the number of valid WARs could be very large, and a user may not be interested in all WARs, but rather a small subset of them. For example, if the inventory of soda becomes too large in a supermarket, the manager may be only interested in the WARs involving soda. It would be desirable if the user can interact with the mining system to obtain a set of interesting WARs by varying the parameter values. Inspired by this, our proposed approach supports both batch mode and interactive mode. In the batch mode, all qualified WARs are returned. On the other hand, in the interactive mode, a user not only can choose a set of interesting frequent itemsets to further investigate, but also can vary the support, interest, and density thresholds, so that only those qualified WARs for the interesting itemsets are returned.

The contributions of this paper are summarized as follows.

- A new class of association rule problem — WAR is proposed.
- Due to the nature of this problem, the mining process is accomplished by a twofold approach: first generating frequent itemsets and then deriving WARs from each frequent itemset.
- During the WAR derivation process,
  - The concept of density is employed to separate transaction concentrated regions from the rest.
  - The geometric properties preserved by the density metric is employed to prune the search space.
  - The weight domain space is partitioned in such a manner that the total number of grids is limited by some parameter N so that the available memory/storage space can be fully utilized.
  - An efficient ordered shrinkage algorithm is proposed to derive WARs from a high density region through shrinkages.
- The computational complexity is analyzed.
- This approach not only supports the batch mode mining but also the interactive mode.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of some related work. The problem is formulated is Section 3 while Section 4 outlines the general approach. Section 5 and 6 present the space partition and WAR generation, respectively. The application of our algorithm to interactive data mining is described in Section 7. Section 8 shows the experimental results whereas the conclusion is drawn in Section 9.

2. Related Work

2.1. Quantitative Association Rule

Association rules with numerical attributes have been studied before. Srikant et. al. (Srikant and Agrawal, 1996a) introduced the problem of mining association rule involving quantitative attributes. The domain of each quantitative attribute is divided into a set of intervals via equi-depth partitioning, i.e., each interval is with a similar number of attribute values. Intervals are combined as long as their support is less than the user specified *max-support*. Each of original intervals and the combined intervals is treated as a different item. Thus, the attribute value of the newly "generated" items is binary: 1 stands for that the original attribute value is within the specified interval; 0 stands for that the original attribute value is not within the specified interval. The algorithms for mining traditional association rules can then be applied on these newly generated binary items. In addition, that paper also presented a measure called "partial completeness" to quantify the information loss due to partitioning and proves that the equi-depth partition minimizes the information loss. *Max-support* is used to restrict the combining of adjacent intervals, but it can also remove some strong and interesting rules, e.g., rules with very high support which is greater than the max-support threshold. The empirical comparison of the quantitative association rule and our proposed weighted association rule is in Section 8.3. Lent et. al. (Lent et al, 1997) described a method for clustering two dimensional association rules. Both of them are variations of the quantitative association rules.

2.2. Optimized Association Rules

In (Fukuda et al, 1996a), Fukuda et. al., introduced so-called "optimized association rules" of the form $U \cap C_1 \Longrightarrow C_2$, which contain a single uninstantiated condition $U = A \in [l, u]$ on the left hand side, and proposed a novel schemes to determine values for variable l and u such that either confidence or support of the rule is maximized while the other satisfies the requirement. The result in (Fukuda et al, 1996a) is then extended to the case where the rules contain two uninstantiated numerical attributes on the left hand side (Fukuda et al, 1996b).

In (Rastogi and Shim, 1998), the problem is again generalized to, and an efficient algorithm is proposed for the case that more than two uninstantiated attributes on the left hand side that can be either categorical or numerical. In addition, Rastogi and Shim further generalized optimized support association rule problem by permitting rules to contain disjunctions over uninstantiated numerical attributes (Rastogi and Shim, 1999). A dynamic programming algorithm is proposed to compute these rules. However, all of these approaches assume that a categorical constraint is prefixed on both sides and the attributes in U are known in advance. On the other hand, our proposed weight association rule model does not assume this restriction.

The optimized gain rule is proposed in (Brin et al, 1999). The gain of a rule $U \cap C_1 \Longrightarrow C_2$ is defined as the difference between its support and the value of $minconf$ times the support of $U \cap C_1$ where $minconf$ is the user specified minimum confidence. A bucketing algorithm is presented to provide a linear solution to mine optimized gain rules containing one numeric attribute. A dynamic programming approach is also devised to approximate the optimized gain sets involving two numeric attributes.

2.3. Ratio Rule

Korn et. al., proposes a new variation of association rule, ratio rule (Korn et al, 1998). "$soda : snacks = 2 : 1$" is an example of a ratio rule, which means that the ratio of the quantity of soda and snacks that a customer purchases at a time is usually 2 to 1. Based on the ratio rules, one can easily answer questions such as "how many soda a customer would buy if he purchased 3 bags of snacks". The algorithm for ratio rule is fast and scalable, requiring only one pass over the dataset. The ratio rule is especially useful if the data points are linearly correlated but not clustered.

2.4. Subspace Clustering

An approach, CLIQUE, is proposed in (Agrawal et al, 1998) to automatically discover dense clusters embedded in subspaces of maximum dimensionality. The space is partitioned into a set of small units and a density threshold τ is specified by users to define the selectivity of a unit. A bottom-up approach is used to find all dense units and adjacent dense units are identified by a depth-first traversal to form clusters. As a follow-on work, an entropy-based algorithm, ENCLUS, is presented in (Cheng et al, 1999). The entropy is used to measure the quality of subspace clusters and can provide efficient pruning of uninteresting subspace combinations. The subspace clustering does not take into account the implication relationship among different attributes, which is taken into account in the weighted association rule model.

3. Problem Formulation

Let $\Im = \{i_1, i_2, \ldots, i_M\}$ be a set of items and $\wp$ be the set of non-negative integers. A pair $\langle x, w\rangle$ is called a **weighted item** where $x \in \Im$ is an item and $w \in \wp$ is the weight associated with x. A transaction is a set of weighted items. Note that an item may appear in multiple transactions and may have different weights in different transactions. For example, a customer purchases 15 items from the fashion department and 10 items from the sport department; while another customer purchases 20 items from fashion department and 5 items from the book department. These two transactions can be represented as $T_1 = \{\langle fashion, 15\rangle, \langle sports, 10\rangle\}$ and $T_2 = \{\langle fashion, 20\rangle, \langle book, 5\rangle\}$, respectively. A triple $\langle x, l, u\rangle$ denotes that the weight associated with the item x is within the range $[l, u]$ where l and u are non-negative integers and $l \leq u$. Note that we can always view a weighted item $\langle x, w\rangle$ as a special case of the interval weighted item $\langle x, l, u\rangle$ where $w = l = u$. Therefore, we will use the term *weighted item* to represent it as well if no ambiguity will occur[1]. Given two weighted items $I_1 = \langle x_1, l_1, u_1\rangle$ and $I_2 = \langle x_2, l_2, u_2\rangle$, we call I_1 a **generalization** of I_2 (and I_2 is a **specialization** of I_1) iff $x_1 = x_2$ and $l_1 \leq l_2 \leq u_2 \leq u_1$. For example, $\langle fashion, 10, 20\rangle$ is a specialization of $\langle fashion, 10, 25\rangle$. Note that any item x can be viewed as a weighted item whose weight is allowed to take any value within the domain $\wp$. Then any weighted item $\langle x, l, u\rangle$ can be view as a specialization of the item x.

We use the term **weighted itemset** to represent a set of weighted items. Let $item(X)$ denote the set of items that are involved in a weighted itemset X, i.e., $item(X) = \{x \mid x \in \Im, \langle x, l, u\rangle \in X\}$. Given two weighted itemsets X_1 and X_2, X_1 is a *specialization* of X_2 (or X_2 is a *generalization* of X_1) iff $item(X_1) = item(X_2)$ and each weighted item in X_1 is a specialization of a weighted item in X_2. For instance, $\{\langle fashion, 10, 20\rangle, \langle book, 5, 7\rangle\}$ is a specialization of $\{\langle fashion, 10, 20\rangle, \langle book, 5, 10\rangle\}$. Given a transaction T and a weighted item $\langle x, l, u\rangle$, we say that T **supports** this weighted item iff there exists a weighted item $\langle x, w\rangle \in T$ such that $\langle x, w\rangle$ is a specialization of $\langle x, l, u\rangle$. Similarly, we say that a transaction T **supports** a weighted itemset X iff T supports each individual weighted item in X. For instance, if $X = \{\langle fashion, 10, 20\rangle, \langle book, 5, 10\rangle\}$,

[1] Indeed, $\langle x, w\rangle$ can be treated as an abbreviation of $\langle x, w, w\rangle$.

then T_2 supports X while T_1 does not. Given a weighted itemset X and a set of transactions, referred to as $\Re$, we say X has **support** s in $\Re$ iff $s\%$ of transactions in $\Re$ support X. Note that the support of a weighted itemset is always less than or equal to the support of any of its generalization.

A **weighted association rule (WAR)** is an implication $X \longrightarrow Y$ where X and Y are two weighted itemsets and $item(X) \cap item(Y) = \emptyset$. A transaction is said to *support* a WAR $X \longrightarrow Y$ iff this transaction supports the weighted itemset $X \cup Y$. In turn, we define the *support* of the WAR as the support of $X \cup Y$. In addition, we say that the WAR $X \longrightarrow Y$ holds in the transaction set $\Re$ with *confidence* c iff $c\%$ of the transactions in $\Re$ that support X also support Y. In other words, the confidence of the WAR is the ratio of the support of $X \cup Y$ over the support of X. We next introduce the concept of *density* which is has been used for clustering in (Agrawal et al, 1998) (Ester et al, 1996) (Ester et al, 1997) (Wang et al, 1997). The density of a WAR is defined as the ratio of the actual support of the WAR and the "expected" support of the WAR. The expected support of a WAR is the support if the weight of each item on each transactions was truly uniformly distributed. We will elaborate on the density definition in Section 5. In this paper, we assume that Y only contains one weighted item for the sake of simplicity. All results presented here can be easily extended to the case that multiple weighted items appear in Y.

Given a transaction set $\Re$, our objective is to find a set of **Weighted Association Rules** (WAR) which have support, confidence, and density greater than or equal to some user-specified minimum support (referred to as *minsup*), minimum confidence (referred to as *minconf*), and minimum density (referred to as d). Since, there could be a huge number of qualified WARs, in this paper, we aim at mining **maximum WAR**. A qualified WAR $X \longrightarrow Y$ is a maximum WAR if for any generalization X' of X and Y' of Y where $X' \neq X$ and $Y' \neq Y$, neither of $X' \longrightarrow Y$, $X \longrightarrow Y'$, nor $X' \longrightarrow Y'$ is a qualified WAR. The fundamental techniques for the interactive mode and batch mode are the same except that some preliminary results are stored to facilitate the final WAR generation in the interactive mode. Thus, in Section 4 to 6, we will present the techniques and address the variations to suit the interactive mode in Section 7.

4. General Approach

As a known fact, there can be a huge number of potential WARs due to the numerical nature of the weight. Efficient pruning of such a huge search space becomes a crucial task in the mining process. Unlike (Srikant and Agrawal, 1996a), we design a twofold approach based on an observation we made: *the support of a weighted itemset is always less than or equal to the support of any of its generalizations*. This indicates that, for any weighted itemset I, its support is always less than or equal to the support of $item(I)$. This suggests that we can first calculate frequent itemsets[2] (without considering the weights) and then examine the weight factor for each frequent itemset to generate the WARs. Thus, we employ the following twofold approach.

[2] A frequent (weighted) itemset is a (weighted) itemset whose support is large than or equal to the threshold *minsup*.

1. Generate frequent itemsets. In this step, we ignore the weight associated with each item in the transaction set.
2. For each frequent itemset, find the WAR(s) that meets the support, confidence, and density thresholds.

Since the first phase is mainly the frequent itemset counting (as in the traditional association rule mining), many algorithms (Agrawal et al, 1993) (Agrawal and Srikant, 1994) (Savasere et al, 1995) (Brin et al, 1997a) (Bayardo, 1998) have been proposed to efficiently accomplish such a task. We will not elaborate it in this paper.

After we obtain the set of frequent itemsets, referred to as F, we examine them to generate the weighted association rules. Given an itemset I of cardinality n, the domain of the weights of all items forms an n dimensional space $\wp^n = \overbrace{\wp \times \wp \times \ldots \times \wp}^{n}$, where each dimension corresponds to the weight of one item. (For the simplicity, we assume the domain of the weight on each dimension to be the same. However, our approach can be easily generalized to the case that has different domains for different items.) Each specialization of I corresponds to an n-dimensional (rectangular) box within this space. Our objective is to find the maximum box(es) so that support, confidence, and *density* are satisfied. To facilitate this process, we discretize the space into a set of grids and divide the second phase into two sub-phases.

1. Space partition and counter generation: The goal is to identify, for each frequent itemset, those (dense) grids that satisfy the density requirement. Efficient pruning technique is provided to reduce the number of grids that need to be evaluated and maintained the counters.
2. Weighted association rules generation: The goal is to generate the maximum WARs from the dense boxes enclosing the adjacent dense grids. As the dense boxes generally do not satisfy the confidence requirement, an ordered shrinkage approach is developed to shrink the dense boxes in an orderly way to meet the confidence requirement.

We elaborate each of them in the following sections.

5. Space Partition and Counter Generation

As mentioned before, we introduce the density concept as a construct to develop effective pruning techniques to identify candidate boxes for WAR mining. The introduction of the density threshold is not only for the implementation of a faster algorithm, but also to ensure that there is sufficient support for a rule in the entire specified regions. It is difficult to tell whether a rule is valid in a region where the density is low (too few evidences). If a rule has high support, but there exists some region with low density, the the rule may be split into two or more rules, each of which contains a high-density region. As a result, no high support rule would be lost due to the density threshold. We want to keep the additional parameters that need to be specified by the users to minimum. The intent is to introduce one density threshold that is applicable to all grids, regardless of the number of dimensions. A straightforward partitioning method is to divide each dimension into a fixed number of partitions. However, under this approach as the number of dimension increases, the number of grids grows exponentially, while the average density of each grid drops rapidly. This implies that different density thresholds would be needed for grids of different dimensionalities. We hence use an alternative approach that keeps the number of grids fixed regardless of the number of dimensions. Although this approach needs one density threshold for pruning, it does create additional complexities on the pruning algorithms as the projection of an n-dimensional grid onto an $(n-1)$-dimensional subspace does not align on the grid boundaries of the $(n-1)$-dimensional subspace.

Given an itemset I of cardinality n, let m_k be the number of intervals, referred to as *base intervals*, where the domain of the kth dimension is partitioned into. For the sake of simplifying the explanation, we assume that each dimension is divided into the same number of base intervals[3], i.e., $m_k = m$, for all $k = 1, 2, \ldots, n$, where m is a positive integer. Then, $N = \Pi_{1 \le k \le n} N_k = m^n$ is the total number of grids within the space where each grid is an n-dimensional cube whose span on each dimension is exactly one base interval. Unlike other space partitioning methods (Srikant and Agrawal, 1996a) (Wang et al, 1997) (Agrawal et al, 1998) which fix the number of intervals on a dimension despite of the dimensionality[4], the total number of grids for each itemset is "fixed" in our approach, i.e, N is regarded as a (large) constant number for every itemset regardless of its cardinality (n). Then, the number of intervals that a dimension is partitioned into, m, is a function of N and the cardinality of the itemset and may be different for different itemsets. Looking ahead, the average number of counters needed for each itemset is linearly proportional to N. One advantage of this partitioning method is that we can easily control the number of counters by varying the value of N. This provides us with the capability to take full advantage of the available storage space dynamically. For an n-itemset, we have $m = \sqrt[n]{N}$. Since m may not be an integer for every possible value of N, instead of enforcing a unique N for all itemsets, we usually set an upperbound $\widehat{N}$ of N according to the space availability and require that for a given n-itemset, N has to be the largest possible value ($\le \widehat{N}$) that makes $\sqrt[n]{N}$ an integer. That is, $N = \left(\lfloor \sqrt[n]{\widehat{N}} \rfloor\right)^n$ which may be slightly different for different itemsets. Figure 1(a) illustrates the space partitions for a two itemset $\{fashion, sport\}$ and a three itemset $\{fashion, book, sport\}$ where $N = 225$. Therefore, each dimension is divided into $\lfloor\sqrt{225}\rfloor = 15$ intervals for 2-imteset $\{fashion, sport\}$ and is divided into $\lfloor\sqrt[3]{225}\rfloor = 6$ intervals for 3-itemset $\{fashion, book, sport\}$. We need to comment that as the dimensionality increases, although the interval on each dimension becomes larger, the average number of transactions in each grid still remains roughly the same.

The grid is taken as the granularity of our WAR mining process. Each grid corresponds to a specialization of I. In addition, any (n-dimensional) *box* within this space, which is the union of a set of adjacent grids and is rectangular in shape, also uniquely corresponds to a specialization of I. For any two weighted itemsets I_1 and I_2, if I_1 is a specialization of I_2, then the box corresponding to I_1 is totally included in the box corresponding to I_2. A grid is the finest box while the domain $\wp^n$ is the largest box we consider during the mining process. All n-dimensional boxes within the domain essentially form a lattice through the

[3] All results presented in this paper can be extended to the case where different partitions are employed for different dimensions.

[4] Note that the number of grids grows exponentially as the dimensionality increases.

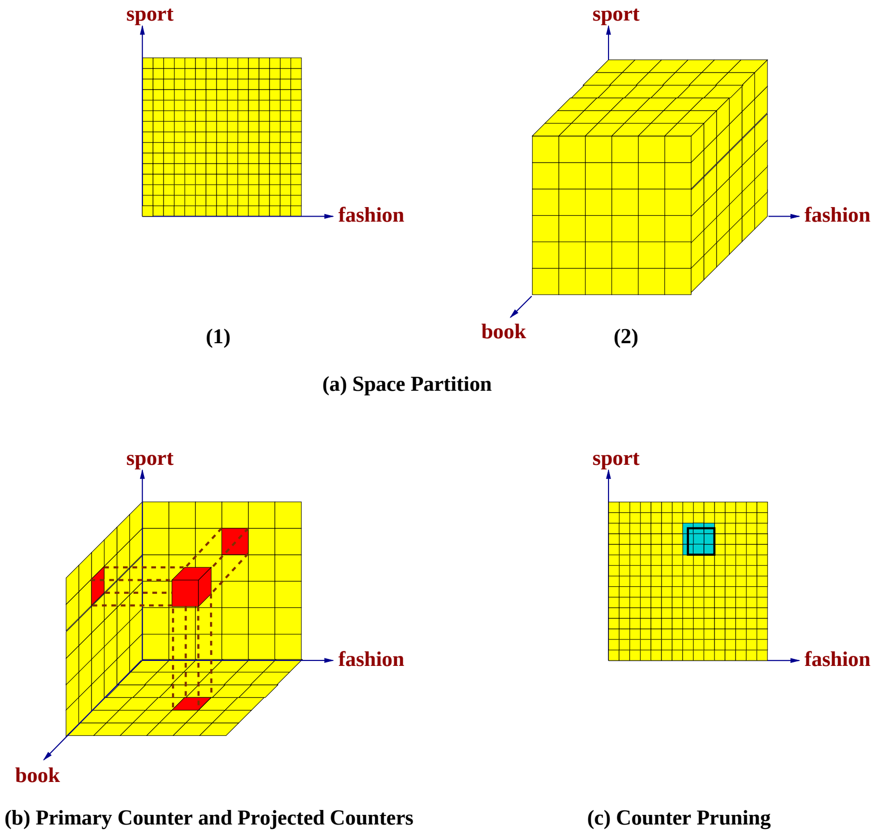

Fig. 1. Partitioning and Counters

enclosure relationship. This partial order among boxes naturally coincides with the specialization/generalization relationship between weighted itemsets that are specialized from the same itemset I. Therefore, in the remainder of this paper, we will use the term "box" and "weighted itemset" interchangeably if no ambiguity will occur. For example, we say that a transaction *supports* a box (within this space) iff this transaction supports the corresponding weighted itemset. In turn, the *support* of a box is defined as the support of the corresponding weighted itemset.

Let α be the average number of transactions which support a grid. We have $\alpha = \frac{|\Re|}{N}$. The *density* of a grid is defined as the ratio of the support of this grid and α. (Note that since α is independent of dimensionality, as long as N and $|\Re|$ is unchanged, α remains constant regardless of the number of dimensionality.) Intuitively, density can be viewed as an indication of relative concentration of transactions within the space. A grid is *dense* iff its density is above a threshold d where d is a small real number. Usually, we require that $d \geq 1$. In order to limit our search space, we only investigate dense grids and the region formed by dense grids. Unlike box, a region does not have to be of rectangular shape. The motivation behind this is that we only want to report WARs which represent significant patterns in the data. Therefore, a range will not be included in a

WAR if there is not enough evidence (density) to support it. A *dense region* is the union of a set of adjacent dense grids. Each box within a dense region, referred to as *dense box*, is a candidate of frequent weighted itemset. The *volume* of a box is defined as the number of distinct grids it contains. By definition, the volume of a grid is 1 while the domain $\wp^n$ has volume N.

Counter Pruning

After partitioning the space, we then collect the counters for each grid. For an n-itemset, there are n possible WAR formats since each item can serve as the right hand side. In order to evaluate the confidence of these WARs, for each grid, we not only need the counter, referred to as *primary counter*, to record the support of this grid, but also need the counters, referred to as *projected counters*, which record the support of the grid projections on each $(n-1)$-dimensional space, i.e., the support of each possible left hand side itemset of the WAR. For example, Figure 1(b) shows the projections of the space partition in Figure 1(a)(2) onto each two dimensional space. For any grid (shown as the dark shaded three dimensional grid in Figure 1(b)) resulting from the space partition in Figure 1(a)(2), a primary counter (recording the support of this grid) and three projected counters (recording the support of the projection of this grid onto each two dimensional space $\{fashion, sport\}$, $\{fashion, book\}$, and $\{sport, book\}$, respectively) are required for the WAR generation.

Although n projected counters are necessary for each grid in order to generate WARs from an n-itemset, they can indeed be shared by many itemsets. For example, $\{fashion, sport, book\}$ and $\{fashion, sport, houseware\}$ have $\{fashion, sport\}$ in common and hence can share the corresponding projected counters (which only need to be generated once). In general, for a frequent n-itemset I, we need to generate two sets of counters. In the first case, since I is a frequent itemset, the counters are needed to derive WARs in the form of $X \longrightarrow Y$ where $X \cup Y = I$. In the second case, the counters are needed to handle $I' = I \cup V$ which is also a frequent itemset and I might serve as the left hand side of the WAR $I \longrightarrow V$ where $V \in \Im$ is an item. In the first case, there are N primary counters need to be generated from an n dimension space. In the second case, projected counters (from an $n+1$ dimensional space) onto the domain space of I need to be generated. Because each dimension is partitioned into $\sqrt[n+1]{N}$ base intervals and the weight domain space of I has n dimensions, the total number of projected counters is $N^{\frac{n}{n+1}}$. Note that counters for the above two cases are both associated with grids of the same weight domain space, i.e., the weight domain space of I. The only difference is that the grids for projected counters (second case) are on a coarser resolution. (Each dimension is divided into $\sqrt[n+1]{N}$ intervals instead of $\sqrt[n]{N}$ intervals as in the first case.) This can be clearly observed in Figure 1. The weight domain space of $\{fashion, sport\}$ is partitioned into 225 grids to collect the primary counters (Figure 1(a)(1)) whereas the projection of the partition of the weight domain space of $\{fashion, sport, book\}$ consists only 36 grids (Figure 1(b)). As a result, each projected grid overlaps with 9 grids partitioned for primary counters (Figure 1(c)).

Since at most $N + N^{\frac{n}{n+1}}$ counters may be required for each frequent n-itemset, the total number of counters for mining all WARs for all frequent itemsets is $O(N \times |F|)$ where $|F|$ is the number of frequent itemsets. It would be preferable if the number of counters can be reduced further. In fact, the density requirement[5] can be used as a pruning tool as we only derive WARs from the dense region and only counters corresponding to dense grids need to be collected. Thus, a levelwise paradigm can be devised for counter pruning based on the following two **geometric properties**.

- An n-dimensional grid b partitioned for primary counter cannot be dense if one of its projection b' onto an $(n-1)$ dimensional space has support less than $d \times \alpha$. For example, in Figure 1(b), if the dark shaded grid on $\{fashion, sport\}$ space has support less than $d \times \alpha$, the dark shaded three dimensional grid cannot be dense.
- For an n-dimensional grid b' partitioned for projected counter, let $\Phi(b')$ be the set of n-dimensional grids that are partitioned for primary counters and overlap with b'. In Figure 1(c), the solid dark line contours b' and the shaded grids forms $\Phi(b')$. b' can not be dense if the support for $\Phi(b')$ is less than $d \times \alpha$.

These two geometric properties suggest that we should collect the projected counters (from n-itemsets) onto $(n-1)$-itemsets before the primary counters for n-itemsets and collect the primary counters for n-itemsets before the projected counters (from $(n+1)$-itemsets) onto n-itemsets. Therefore, the counters for all frequent n-itemsets are mapped to two levels: Level $2(n-1)$ and $2(n-1)+1$. Figure 2 illustrates the process to generate counters for itemset up to $\{fashion, sport, book\}$. The arrow indicates the pruning direction. We do not collect the primary counters for 1-itemset because a WAR involves at least two items.

After generating necessary counters, we also find all dense grids. Then, dense regions, which serve as the basis for WAR generation, can be easily identified by a depth-first traversal through neighboring dense grids similar to (Agrawal et al, 1998).

6. WAR Generation

Some user may be interested in all qualified WARs for a given dataset while others may want only a few WARs with largest volume. We will show, later in this section, that two different data structures are utilized to optimize the performance for each case. Given an n-itemset $I = \{i_1, i_2, \ldots, i_n\}$, there are potentially n different WAR formats because each item might serve as the right hand side. In order to search for the range associated with each item, we start from the minimum bounding box[6] of each dense region and shrink towards the maximum WARs. An alternative algorithm is to pick a grid and grow towards maximum WARs. Since the maximum WARs usually have large volumes, shrinking towards a maximum WAR is generally more efficient than growing towards it. To the best of our knowledge, this is the first algorithm to use shrinking instead of growing in mining association rules or clustering. A *shrinkage* is defined as the action that reduces the span of a box over one dimension by exactly one base interval. Without loss of generality, we assume that there is only one dense region for

[5] In order to be a dense grid, a grid has to have support at least $d \times \alpha$.
[6] A bounding box of a dense region is a box that contains this dense region. The minimum bounding box is the one with smallest volume among all bounding box for this dense region.

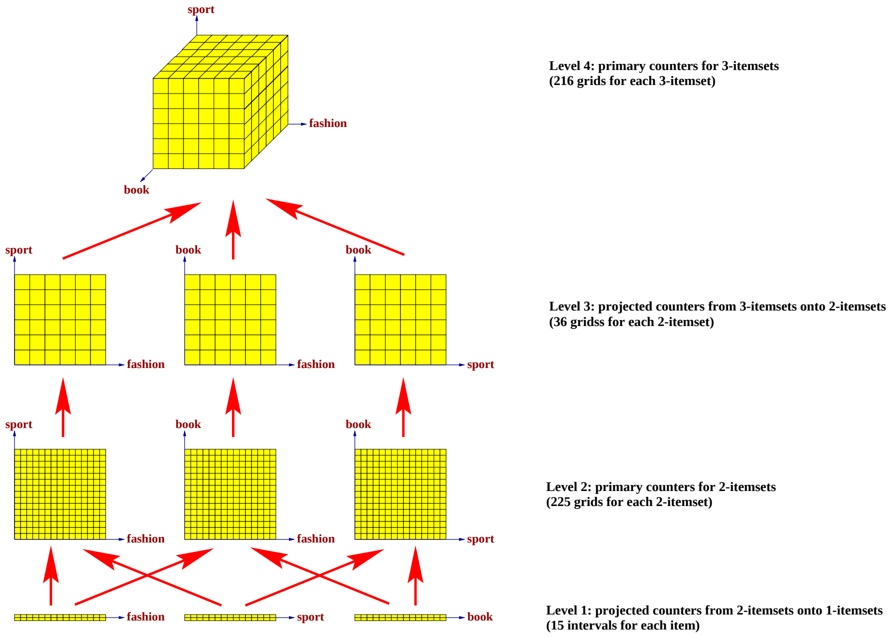

Fig. 2. Counter Pruning

the following explanation. An example is illustrated in Figure 3. Figure 3(a) is a dense region whose minimum bounding box is shown in Figure 3(b) where the dark shaded grids and light shaded grids are dense grids and non-dense grids, respectively. For each item, the range of its weight can be shrunken towards two directions: increasing the lowerbound or decreasing the upperbound. Thus, there are $2n$ different shrinkages applicable to a weighted n-itemset. As a result, $2n$ new weighted n-itemsets can be generated. Note that these newly generated n-itemsets are all specializations of the original one. Since these new weighted item sets are produced by one shrinkage from the original weighted itemset, we call them the *immediate specializations* of the original weighted itemset. The original weighted itemset is an *immediate generalization* of these new weighted itemset. Therefore, each weighted itemset (except those that represent single grid on some of its dimensions) has $2n$ immediate specializations. On the other hand, each weighted itemset except the one that represent the whole domain has $2n$ immediate generalizations. Figure 3(c) shows the six different directions to shrink a weighted 3-itemset involving $\{fashion, sport, book\}$ and their corresponding outcomes. It is obvious that all of these generated weighted 3-itemsets are immediate specializations of the original box shown in Figure 3(b).

Clearly, each dense box can be reached by a shrinkage from its immediate generalization(s) and hence is reachable by a set of shrinkages from the minimum bounding box of the dense region. Assuming that we can only perform one shrinkage on one candidate box at a time, the entire process can be viewed as a sequence of operations (B_j, H_j) $(j = 1, 2, \ldots)$ where B_j is a candidate box and H_j is a shrinkage along some direction. Let Ψ_j be the set that includes the minimum bounding box of the dense region (B_1) and all boxes generated via operations $(B_1, H_1), \ldots, (B_{j-1}, H_{j-1})$. Ψ_j thus represents the set of candi-

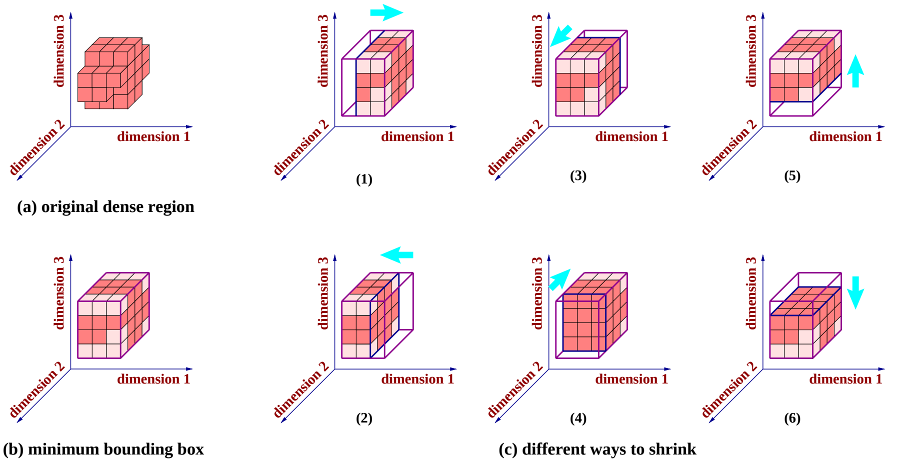

Fig. 3. Region shrinkage

date boxes (for further shrinking) available before the jth operation. Clearly, we have $B_j \in \Psi_j$ at each step. The sequence stops when all WARs are generated. For the box B_j visited at the jth step, there exists a subsequence of operations $(B_{j_1}, H_{j_1}), (B_{j_2}, H_{j_2}), \ldots, (B_{j_r}, H_{j_r})$ such that B_{j_1} is the minimum bounding box of the dense region and B_{j_k} is the box produced by performing shrinkage $H_{j_{(k-1)}}$ on $B_{j_{(k-1)}}$ where $1 < k \le r$ and $1 \le j_1 < j_2 < \ldots < j_r < j$. $H_{j_1}, H_{j_2}, \ldots, H_{j_r}$ is a *shrinkage path* to B_j from the minimum bounding box of the dense region.

Note that, at each step, there are multiple candidate boxes in Ψ_j and different shrinking directions to choose from. As a result, different algorithms for picking candidate box and shrinking direction can produce different sequences of operations and hence could require different number of operations before all necessary WARs are generated. Therefore, the efficiency of an algorithm depends on the amount of time consumed at each operation and the number of operations needed.

Since each weighted itemset I has $2n$ immediate generalizations, there are $2n$ different weighted itemsets can generate I by one shrinkage. One possible approach is the brute force algorithm. Let Ψ'_j be Ψ_j excluding those boxes which satisfy either of the following conditions: (1) all of its immediate specializations are in Ψ_j, or (2) it is a maximum WAR. At each step, the brute force algorithm randomly picks a box from Ψ'_j. The algorithm terminates when $\Psi'_j = \emptyset$. The computational complexity of the brute force algorithm is $O\left((2n)^{n \times \sqrt[n]{N}}\right)$. Therefore, this approach is indeed inefficient. Intuitively, this inefficiency is caused by the fact that a box may be visited multiple times via different shrinkage paths. In other words, there exist operations in the sequence (B_j, H_j) and $(B_{j'}, H_{j'})$ $(j \ne j')$ which produce the same box. This would lengthen the operation sequence and cause inefficiency. An example is shown in Figure 4. There are two paths to reach the box in Figure 4(d) from the box in Figure 4(a). The small arrows indicate the shrinking directions adopted at each step. In this figure, the shrinkage path "(a) $\rightarrow$ (b) $\rightarrow$ (d)" is "reducing the upper bound of dimension 2" followed by "reducing the upper bound of dimension 1", while the shrinkage path "(a) $\rightarrow$ (b) $\rightarrow$ (d)" is "reducing the upper bound of dimension 1" followed

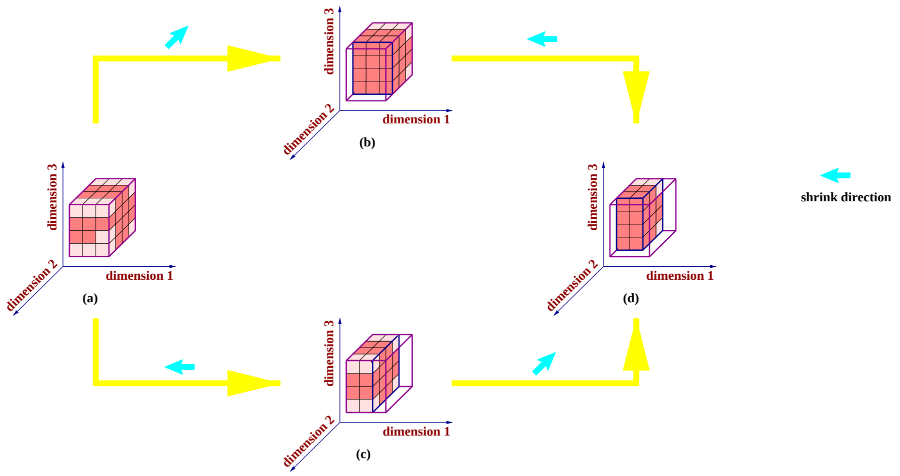

Fig. 4. Multiple shrinkage paths

by "reducing the upper bound of dimension 2". The set of shrinkages of the two paths is the same. The only difference is that different paths adopt a different permutation of these shrinkages. In general, if it takes b shrinkages to reach from box B' from B (where B' is enclosed within B), then there exist $b!$ different shrinkage paths. In order to eliminate the redundancy of weighted itemset generation, we introduce an **ordered shrinkage** technique. This guarantees that each weighted itemset is generated exactly once.

6.1. Ordered Shrinkage

We first choose a permutation, referred to as Ω, of $2n$ different shrinking directions[7] and retain this order during the entire process. For example, the six shrinking directions in Figure 5(c) can be ordered as *increasing lowerbound of dimension 1, decreasing upperbound of dimension 1, increasing lowerbound of dimension 2, decreasing upperbound of dimension 2, increasing lowerbound of dimension 3, decreasing upperbound of dimension 3.* Then, during the shrinking process, a shrinkage of the kth direction in the order sequence Ω can be performed on a box B only if no shrinkage of direction after the kth one in the ordered sequence has been performed to generate B. We call such shrinkage a *valid* shrinkage. For instance, if we take the box in Figure 3(c)(4) as the candidate to generate new weighted itemsets (also shown in Figure 5(a)), according to the order we pick, three shrinking directions can be applied as illustrated in Figure 5(b). Figure 5(c) shows an invalid shrinkage because decreasing the upperbound of dimension 1 is not allowed after decreasing the upperbound of dimension 2. Note that the box shown in Figure 5(c) can be obtained by a valid shrinkage from another immediate generalization (in Figure 3(c)(2)) of it as shown in Figure 5(d).

It is obvious that this ordered shrinkage approach completely eliminates the

[7] There are $(2n)!$ different permutations. Here, we randomly choose one of them.

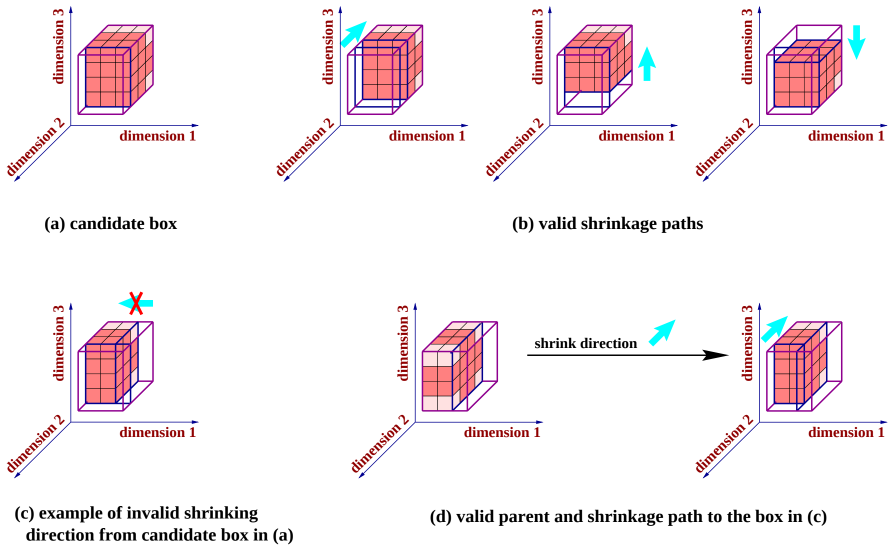

Fig. 5. Ordered shrinkage

redundancy of box generation in the previous brute force approach by restricting the possible directions a box can be shrunken at any stage. In fact, redundancy will occur only if there exist multiple shrinkage paths to a certain weighted itemset. The ordered shrinkage successfully avoids this redundancy by providing each possible weighted itemset a unique shrinkage path from the root. The computational complexity of the ordered shrinkage for one itemset is $O(N^2)$.

6.2. Proof of Correctness

Lemma 6.1. For any two weighted itemsets WI_1 and WI_2, where WI_1 is a specialization of WI_2. WI_1 can always be generated by performing a sequence of ordered shrinkages on WI_2.

Proof. Let $(l'_1, u'_1), (l'_2, u'_2), \ldots, (l'_n, u'_n)$ and $(l_1, u_1), (l_2, u_2), \ldots, (l_n, u_n)$ are the coordinates of WI_1 and WI_2 on each dimension, respectively. Then the following way can shrink WI_2 to WI_1: first shrink l_1 to l'_1, then u_1 to u'_1, then l_2 to l'_2, $\ldots$, u_n to u'_n. Thus this lemma holds. $\square$

Lemma 6.2. For any two weighted itemsets WI_1 and WI_2, where WI_1 is a specialization of WI_2. There exists exactly one valid shrinkage path from WI_2 to WI_1.

Proof. Let's assume the contradiction is true. There exist two paths (P and Q) to shrink from WI_2 to WI_1. Let's assume the first difference is that at a step, P shrinks the bound on p whereas Q shrinks the bound q and $p < q$, then after this step, in Q, p can not be shrunk any longer with respect to the order Ω. As a result, Q and P could not shrink to the same box. Thus, the lemma holds. $\square$

6.3. Data Structures and Algorithm

If all maximum WARs are required by the user, a queue can be utilized to maintain the candidate box set (Ψ). We take out the box at the head of the queue for examination and after shrinking, the new boxes are simply appended at the end of the queue. The algorithm terminates when all maximum WARs satisfying the support and confidence thresholds are found.

An immediate optimization would be that whenever some qualified WARs are derived from a box, we will discard it from the candidate box set and will not perform shrinkage on this box any more. In addition, when all necessary shrinkages have been performed on a box, this box will not be maintained in the candidate box set Ψ any longer so that Ψ will be more compact and more efficient. An easy way to achieve this is to pack all necessary shrinkages that can be performed on a box together in the operation sequence. The formal description of the algorithm is in Algorithm 6.1.

Algorithm 6.1. Find Maximum WARs that satisfy support and confidence thresholds

```
MAX_WAR (s, c)
{
   Ψ ← ∅
   Insert(Ψ, { minimum bounding box of the dense region })
   number_found ← 0
   while (Ψ ≠ ∅)
      R ← Pick(Ψ) /* Take a box R from Ψ. */
      if (support(R) ≥ s and confidence(R) ≥ c) then
      /* Both support and confidence of R satisfy the thresholds. */
         OUTPUT ← Append(OUTPUT, R)
         /* Append R to the OUTPUT list */
         number_found ← number_found + 1
      else if (support(R) ≥ s and confidence(R) ≤ c) then
      /* Support satisfies the threshold while confidence does not. */
         NewBoxes ← Ordered_shrink(R, Ω)
         /* Ordered_shrink returns a set of boxes which are shrunk */
         /* from R with the order Ω. */
         Insert(Ψ, NewBoxes)
         /* Insert all boxes in NewBoxes into the heap Ψ */
   return OUTPUT
}
```

However, if the user only needs a number (say, g) of maximum WARs with largest volume, we can choose a different data structure depending on whether g is a relatively small number (comparing to the number of all qualified maximum WARs). If g is a small number, it is not necessary to find all qualified maximum WARs. A possible optimization of the general algorithm would be to pick the box with the largest volume in the candidate box set Ψ to examine at each step j. A heap structure would come handy to facilitate such a process. However, there is a certain amount of overhead to maintain the heap. In turn, due to the *RANDOMIZED-SELECT* algorithm in (Cormen et al, 1990), it would be more efficient to still use the queue structure when g is large. After we find all valid maximum WARs, the *RANDOMIZED-SELECT* algorithm[8], can be used to locate the maximum WAR with the gth largest volume. Then, a linear scan can determine all maximum WARs with volume larger than the gth largest volume. The complexity of these two algorithms is discussed next.

6.4. Complexity Analysis

In the WAR generation phase, the complexity depends upon the number of distinct dense boxes. In the worst case, all grids are dense in the entire space. In our scheme, we keep the number of grids N as a constant regardless of dimensionality. Since each dimension is divided into m base intervals, there are total $m+1$ endpoints on each dimension. For any box, its coordinates on dimension k will be two of these endpoints. As a result, there are total $\binom{m+1}{2}$ different choices on dimension k, which is $\frac{(m+1)\times m}{2}$. The total number of distinct boxes (NB) in an n-dimension space is

$$\begin{aligned} NB &= \tfrac{(m+1)^n \times m^n}{2^n} \\ &= \tfrac{(m+1)^n}{2^n} \times N \end{aligned}$$

Assuming that there are the same number of intervals on each dimension, i.e., $m = N^{1/n}$, then we have

$$NB = \left(\frac{N^{1/n}+1}{2}\right)^n \times N = O(N^2) \tag{1}$$

Therefore, the overall complexity for finding qualified maximum WARs for a given itemset is $O(N^2)$ if a queue structure is used. On the other hand, if a heap structure is utilized, the complexity is $O(N^2 \log N)$. (The complexity of maintaining a heap is logarithmic respect to the number of elements in the heap.) Although the algorithm with queue has lower complexity than the one with heap, it is not necessary that the one with queue always performs better. We will analyze the performance of these two data structures at Section 8.2.

Now, we are examining the effects of the dimensionality by analyzing the derivative of NB with respect to n.

$$\frac{\partial NB}{\partial n} = -\ln\left(\frac{N^{1/n}+1}{2}\right) \times \left(\frac{N^{1/n}+1}{2}\right)^n \times \left(\frac{N^{1/n} \times \ln N}{2n^2}\right) \times N$$

Here we can show that NB decreases with higher dimensionality because the derivative is negative. Moreover, with the larger number of overall grids, NB decreases in a faster pace. Figure 6 shows the total number of distinct boxes with different N values and different dimensions, and it coincides with our conclusion.

[8] The complexity of *RANDOMIZED-SELECT* algorithm is linear with respect to the number of satisfied boxes.

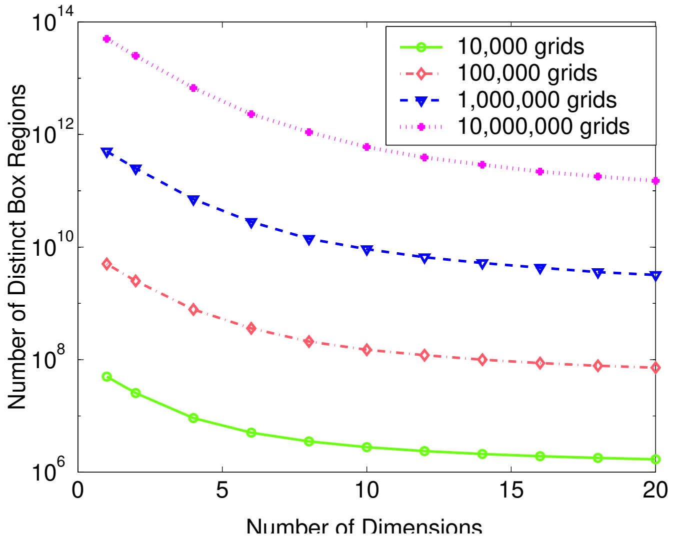

Fig. 6. Number of Box Regions

7. Interactive WAR Mining

During recent years, the interactive data mining becomes more important (Ng et al, 1998) (Hidber, 1999) (Lakshmanan et al, 1999). In this section, we discuss the interactive approach to mine WARs. In the interactive mode, both frequent itemsets and the corresponding grid counters (with respect to some *minsup* and N) are generated offline ahead of time and stored in the secondary storage space, e.g., disks. (The only online computation is the WARs generation by shrinkage. From our experiments, it takes between 1 and 10 seconds to compute WARs for a given frequent itemset if $N = 1,000,000$.) During the interactive session, a user can choose his interested frequent itemset(s) to explore further, set his own support/confidence thresholds, and/or change to a different grid resolution (i.e., the value of N). Since these parameter settings are not available when the offline computation is performed, the values of $minsup_{offline}$ and $N_{offline}$ has to be set in such a manner so that online regeneration of frequent itemset (with respect to some lower *minsup* value) and online repartition weight domain space (into finer grids) can be avoided as much as possible with some storage space constraint.

Let $F_{offline}$ be the set of frequent itemsets with respect to $minsup_{offline}$. There are overall $O(|\ F_{offline}\ | \times N_{offline})$ counters need to be stored and each counter occupies constant space, e.g., 4 bytes. Then the space limitation can be expressed as $|\ F_{offline}\ | \times N_{offline} \leq MAX\_CCOUNTER$. Indeed, two levels of satisfaction are involved: support satisfaction and resolution satisfaction. The *support satisfaction* S_{sup} is defined as the probability that the user specified *minsup* is at least $minsup_{offline}$. If the user specified *minsup* is greater than or equal to *minsup*, then the set of frequent itemsets $F \subseteq F_{offline}$. As a result, F can be trivially computed by filtering out itemsets whose supports are less than *minsup* from $F_{offline}$. Otherwise, we have to recourse to the raw data set to generate itemsets whose supports are between

$minsup$ and $minsup_{offline}$. Obviously, this would delay the WAR generation significantly and hence should be avoided if possible. Intuitively, the relationship among $minsup_{offline}$, $\mid F_{offline} \mid$, and S_{sup} should be as following: S_{sup} is a monotonically increasing convex function of $\mid F_{offline} \mid$ which itself is a exponentially decreasing function of $minsup_{offline}$. This suggests that decreasing $minsup_{offline}$ would improve S_{sup}. The second level is *resolution satisfaction* S_{res} which is the probability that the user required granularity is the same as or coarser than that adopted during the offline computation. Due to a similar reason mentioned above, we want to improve S_{res} if possible. One would expect that S_{res} should be a monotonically increasing convex function of $N_{offline}$. This indicates that S_{res} will be improved with larger $N_{offline}$. Assume that S_{res} and S_{sup} are independent. The overall satisfaction ratio can be expressed as $S_{res}(N_{offline}) \times S_{sup}(\mid F_{offline} \mid)$. We want to maximize it with the constraint that $\mid F_{offline} \mid \times N_{offline} \leq MAX\_CCOUNTER$. We can approximate these functions by past experiences if the user behavior does not change.

8. Experimental Results

We implemented our algorithm in C program language and is executed on an IBM AIX RS6000 machine with a 333 MHz CPU. The data resides on a disk with approximate 8 MB/s throughput. We will explain how the input data is generated in the following section.

8.1. Input Data Generation

To analyze the performance of our proposed approaches, we generate the input data in a similar manner as in (Agrawal and Srikant, 1994). An example of the target environment is on web page recommendation where people tends to access sets of web pages together with certain frequency. A transaction is the web pages visited by a user in a session[9]. For example, a user may visit NFL related web pages 10 times and NBA related web pages 20 times in a session. Here, "NFL[5, 15]" and "NBA[10, 20]" may form a frequent weighted itemset. The transaction size (i.e., number of items in a transaction) is usually cluster around a mean and a few transactions have many items. Typical size of frequent weighted itemsets are also clustered around a mean, with a few frequent itemsets having a large number of items. To create a dataset, our synthetic data generation program takes the parameters in Table 1.

The *volume* of a weighted itemset is measured as the number of grids which contain by the given frequent weighted itemset. We first generate the frequent itemset F. The frequent set generation step is similar to that used for frequent itemset generation in (Agrawal and Srikant, 1994). We chose β to represent the average number of weighted itemsets for a given frequent itemset. Based on F and β, the weighted frequent itemset FW is generated in the following manner. We assume that for a given frequent itemset, the number of frequent weighted itemset is geometrically distributed with parameter q. Then the expected number of frequent weighted itemsets for a give frequent itemset is $\frac{q}{1-q} = \beta$. By solving

9 There are many ways to qualify a session. In general, session can be a period of time varying from hours to days.

[ht]

$\|\Re\|$	Number of transactions
$\|T\|$	Average number of items in a transaction
$\|F\|$	Average number of maximal potentially frequent itemsets
$\|I\|$	Average number of items in a maximal potentially frequent itemset
β	Average number of frequent weighted itemsets for a given frequent itemset
v	average volume of a frequent weighted itemset
M	number of items
N	total number of grids

Table 1. Parameters

this equation, we can know $q = \frac{\beta}{1+\beta}$. For a given frequent itemset I, we generate a random number between 0 and 1. If the random number is less than q, then a new weighted itemset should be generated for I. We assume that the volume of frequent weighted itemsets is normally distributed with mean equal to v. We generate a box for the frequent weighted itemset accordingly. The center of a region is randomly chosen. This process continues until the generated random number is greater than q. We perform this process for all frequent itemsets.

After obtaining the set of frequent weighted itemsets, we embed these itemsets into the transactions in a manner similar to (Agrawal and Srikant, 1994). The transaction size follows a Poisson distribution with mean equal to $|T|$. Each transaction is assigned a series of frequent weighted itemsets. If the itemset on hand does not fit in the transaction, the itemset is put in the transaction with probability 0.5, and the itemset is moved to the next transaction with probability 0.5.

In real world, the frequent weighted itemsets are not "perfectly" contained by transactions. There are two sources of imperfection, one is that the frequent itemsets are not always present together in a transaction (Agrawal and Srikant, 1994). The second imperfection comes from the weight may not always be in the specified range. To preserve the imperfection of data, we intentionally corrupt the data with a corruption parameter γ. When adding an itemset into a transaction, we keep alternating the itemset as long as a uniformly distributed random number between 0 and 1 is less than γ. If the random number is less than γ, one item will be removed in half the cases. For the rest of the cases, the weight of one item is randomly assigned a value outside the specified interval. The corruption parameter for an itemset is fixed and is obtained from a normal distribution with mean 0.5 and variance 0.1.

In this paper, we are most interested in the WAR generation phase of the algorithm, which includes generating counters for a given frequent itemsets and deriving maximum WARs based on these counters. Hence, we generate the data by setting $|\Re| = 10M$(million), $T = 15$, $|F| = 2000$, $M = 10000$, and $|I| = 6$. We choose three values for β: 0.5, 2, 8; two values for N: 100K and 1 M; and two values for v: 50, 500. All the data sets have the same size: 300MB. Table 2 summarizes the data parameter setting.

8.2. Queue vs. Heap

As described in the previous section, one of two data structures, i.e., queue and heap, that may be employed for maintaining the candidate box regions for search. Here, we are examining the pros and cons of these data structures. Since

[ht]

Name	β	N	v
$\beta(0.5).N(100K).v(50)$	0.5	100K	50
$\beta(0.5).N(1M).v(500)$	0.5	1M	500
$\beta(2).N(100K).v(50)$	2	100K	50
$\beta(2).N(100K).v(500)$	2	100K	500
$\beta(2).N(1M).v(50)$	2	1M	50
$\beta(2).N(1M).v(500)$	2	1M	500
$\beta(8).N(100K).v(50)$	8	100K	50
$\beta(8).N(1M).v(500)$	8	1M	500

Table 2. Parameter settings

the frequent itemset generation phase of the algorithm is the same, we focus our attention on the WAR generation phase of the algorithm. In this set of experiments, we set the support threshold to be 0.05, confidence threshold to be 0.3, and density threshold to be 1.5. As explained before, there could be a very large number of valid rules for a give frequent itemset. Usually users are interested in the rules which have the largest volumes. Hence, the algorithm will return a certain number of rules with maximum volume while still satisfying support and confidence thresholds. We examined the algorithm for finding WARs for 2, 3, 4-frequent itemsets, and found that the relative performance is similar, hence we only show the average performance for generating WARs from a 3-frequent itemset in Figure 9. The average performance is obtained by executing the algorithm for 20 different 3-itemsets and is based on the execution time.

From Figure 9, we can see that when the number of rules needed is small, heap structure usually outperforms queue structure because the heap structure can terminate earlier when the necessary number of rules with largest volume have been found. However, when the number of rules needed increases, the queue structure gradually outperforms heap due to less complexity. In the following sections, we are examining the contributions of the parameters β, N, and v to the performance of our algorithm.

The average number of frequent weighted itemsets for a given frequent itemset, β, is an important parameter in the performance of our algorithm. In Figure 9, when β increases (from $\beta(0.5).N(100K).v(50)$ to $\beta(2).N(100K).v(50)$ to $\beta(8).N(100K).v(50)$, from $\beta(0.5).N(1M).v(500)$ to $\beta(2).N(1M).v(500)$ to $\beta(8).N(1M).v(500)$), the benefit of employing heap structure increases because the algorithm may terminate earlier. On the other hand, the performance of the algorithm with queue does not vary significantly with β because it has to search almost all distinct boxes any way.

When the number of grids for each frequent itemset, N, increases from $\beta(2).N(100K).v(50)$ to $\beta(2).N(1M).v(50)$ and from $\beta(2).N(100K).v(500)$ to $\beta(2).N(1M).v(500)$, the performance of the algorithm with either structure slows down. However, the relative performance remains the same.

When the average volume of a frequent weighted itemset, v, increases (from $\beta(2).N(100K).v(50)$ to $\beta(2).N(100K).v(500)$ and from $\beta(2).N(1M).v(50)$ to $\beta(2).N(1M).v(500)$), the benefit of heap is more significant because a frequent weighted itemset can be detected earlier and the algorithm can terminate sooner.

The queue structure usually results in a more or less stable performance, while the performance of the algorithm with the heap structure varies more significantly with the change of data characteristics and the number of rules that a user wants. In conclusion, for a given frequent itemset, if the number of needed rules is small comparing to the overall number of existing weighted frequent itemsets, then a heap data structure should be employed to determine which distinct box should be examined next. For example, if there are 20 weighted frequent itemsets for frequent itemset $\{fashion, sport\}$ and the user only wants 2 rules, then a heap structure should be utilized. Otherwise, a queue structure should be employed.

8.3. Weighted Association Rules vs. Quantitative Association Rules

As explained before, quantitative association rule (QAR) has been proposed for the general numerical attribute values that are associated with every database record. In this paper, we are only interested in the case where each item is associated with a weight; but only a small number of items appear in each transaction. As a result, certain techniques can be used to efficiently mine the WARs, e.g., we can first find frequent itemsets without considering of weights, then we can find rules associated with each frequent itemset. However, our proposed techniques will not work for the case where the general numerical attributes such as age and salary appear in every record or transaction. On the other hand, QAR is the only other approach which we know that may be generalized to mine WARs. Therefore, we compare our approach with the QAR approach.

In essence, the quantitative association rule approach (Srikant and Agrawal, 1996a) first quantizes the domain of every item, and map it into a binary value, where 1 stands for the value of the item does appear in that interval and 0 stands for the value does not appear in that interval. Since at the beginning, it is unknown that which items will appear in an itemset together, all attribute domains have to be partitioned. If a numerical attribute is partitioned into ρ intervals, then these intervals are mapped into $\frac{\rho^2+\rho}{2}$ items because each of the original intervals and the combined intervals is treated as a different item. For each synthetic dataset generated above, if the weight domain of each weighted item is partitioned into 20 intervals, then there are over 2 million binary items for the QAR approach. The QAR approach utilizes a parameter $maxsup$ to reduce the resulting set of rules, i.e., if the support of a rule is larger than $maxsup$, the rule is pruned from the result. It is clear that $maxsup$ can improve the performance, but could potentially degrade the quality of the results significantly. Both parameters $maxsup$ and ρ in QAR affect the quality and performance dramatically. To analyze the performance and quality of these two approaches, we set $\rho = 20$ and $maxsup = minsup + 0.2$ for QAR[10]. For the WAR approach, since it is required to return all valid rules for all frequent itemsets, the queue data structure is utilized, and the density d and the total number of grids (N) for each itemset are set to 1.5 and 1 million, respectively.

The quality of the resulting rules is analyzed via two measurements: recall and accuracy. Since the datasets are generated artificially, we know the set R of all existing rules. First, we compute the volume[11] of each rule $r \in R$, denoted by $V(r)$. For any rule set R', the recall of R' with respect to R is computed as

[10] The rationale behind the parameter setting is that if the values are larger, then the average execution time will be too long.

[11] Other parameters, e.g., support, can be used to calculate the recall and accuracy values.

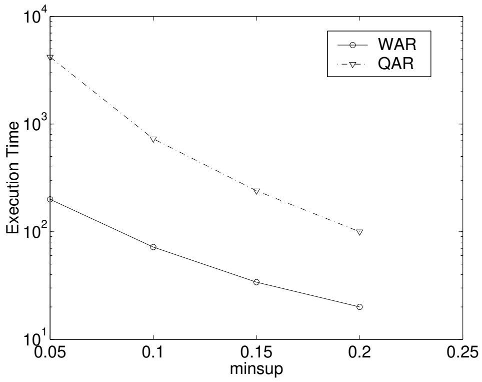

Fig. 7. Execution Time

follows. For each $r \in R$, find the rule $r' \in R'$ such that the overlap of r and r' is the largest, i.e., $V(r \cap r')$ is the largest. Next, the individual recall of R' with respect to r is computed as $\frac{V(r \cap r')}{V(r)}$. The overall recall of R' with respect to R is the average of the individual recalls for each $r \in R$. With a similar manner, the overall accuracy of R' with respect to R is calculated as follows. For each $r' \in R'$, the $r \in R$ which maximizes $V(r \cap r')$ is chosen and $\frac{V(r \cap r')}{V(r')}$ is the individual accuracy of r' with respect to R. The average of all individual accuracy of $r' \in R'$ is treated as the overall accuracy of R' with respect to R. The recall and accuracy of WAR and QAR are analyzed in the following subsections. Since the recall and accuracy values for the eight datasets are very similar, we only present these values for the data set $\beta(2).N(1M).v(500)$. Because the execution time of QAR is very long for the entire dataset, only the first 100K transactions are chosen from the dataset as the input to examine the two approaches.

We use the average execution time of the program to measure the performance. Figure 7 shows the average execution time of the two approaches with respect to *minsup*. The execution time of the WAR approach increases exponentially with decrease of *minsup* because the complexity of the WAR approach is linear to the number of frequent itemsets which usually increases exponentially with smaller *minsup*. However, the QAR approach increases on a much faster pace than that of WAR because the number of counters needed becomes too large. As a result, WAR on average is a couple order of magnitude faster than that of QAR. (Note that the Y-axis is in log-scale.)

Figure 8 shows the recall ratio of the two algorithms with respect to the minimum support threshold. WAR has a significant higher recall value than QAR because the a significant amount of important rules are pruned by the *maxsup* threshold in QAR.

Finally, we examine the accuracy of the two approaches. The weight domain of each item in WAR is partitioned into much finer grids than that in QAR for 2, 3, and 4-itemsets for which most rules exist. As a result, with the finer partition,

Since the recall and accuracy values calculated via support are similar to those calculated via volume, only the values calculated via volume are presented in this paper.

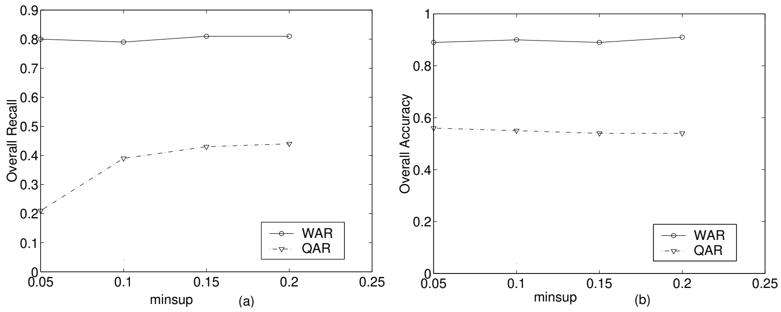

Fig. 8. Quality of the two approaches

the WAR approach produces much more accurate rules than the QAR approach as shown in Figure 8.

9. Conclusion

In this paper, we have investigated a new class of interested problem: weighted association rules, which have wide implications. We proposed a two-fold approach, where the frequent itemsets are first generated (without considering weight) and then the maximum WARs are derived using an "ordered" shrinkage approach. Experimental results show that our proposed approach not only outperforms direct generalization of previous work with order of magnitude, but also produces better quality results.

References

Agrawal R, Imielinski T, and Swami A (1993) Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD Conf. on Management of Data.*

Agrawal R and Srikant R (1994) Fast algorithms for mining association rules. *Proc. 20th Int. Conf. on Very Large Databases.*

Agrawal R, Gehrke J, Gunopulos D, and Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining application. *Proc. ACM SIGMOD Conf. on Management of Data*, pp. 94-105.

Aumann Y and Lindell Y (1999) A statistical theory for quantitative association rules. *Proc. ACM SIGKDD Conf. on Knowlwdge Discovery and Datamining*, 261-270.

Balabanovic M and Shoham Y (1997) Fab: content-based, collaborative recommendation. *Communication of ACM*, 40(9), 64-72.

Bayardo RJ (1998) Efficiently mining long patterns from databases. *Proc. ACM SIGMOD Conf. on Management of Data*, pp. 85-93.

Brin S, Motwani R, Ullman J, and Tsur S (1997) Dynamic Itemset counting and implication rules for market basket data. *Proc. ACM SIGMOD Conf. on Management of Data.*

Brin S, Motwani R, and Silverstein C (1997) Beyond market baskets: generalizing association rules to correlations. *Proc. ACM SIGMOD Conf. on Management of Data.*

Brin S, Rastogi R, and Shim K (1999) Mining optimized gain rules for numeric attributes. *Proc. ACM SIGKDD Conf. on Knowlwdge Discovery and Datamining*, pp. 135-144.

Cheng C, Fu A, and Zhang Y (1999) Entropy-based subspace clustering for mining numerical data. *Proc. ACM SIGKDD Conf. on Knowlwdge Discovery and Datamining*, pp. 84-93.

Cormen T, Leiserson C, and Rivest R (1990) *Introduction to Algorithms*, MIT Press.

Ester M, Kriegal HP, Sander J, and Yu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD).*

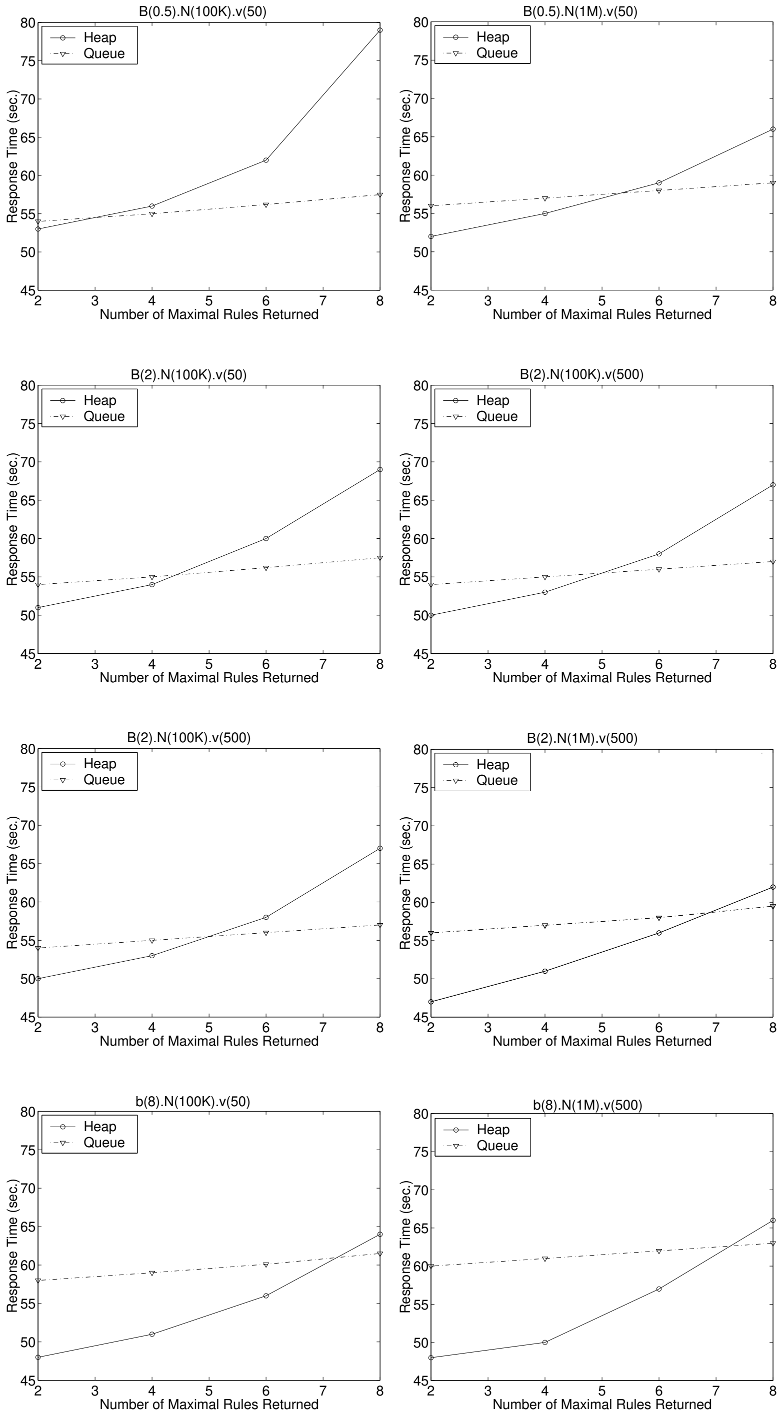

Fig. 9. Execution time of phase two

Ester M, Kriegal HP, Sander J, and Yu X (1997) Density-connected sets and their application for trend detection. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD).*

Fukuda T, Morimoto Y, Morishita S, and Tokuyama T (1996) Mining optimized association rules for numeric attributes. *Proc. 15th ACM Symposium on Principles of Database Systems.*

Fukuda T, Morimoto Y, Morishita S, and Tokuyama T (1996) Data mining using two dimensional optimized association rules: scheme, algorithms, and visualization. *Proc. ACM SIGMOD Conf. on Management of Data.*

Han J and Fu Y (1995) Discovery of multiple-level association rules from large databases. *Proc. 21st Int. Conf. on Very Large Databases.*

Hidber C (1999) Online association rule mining. *Proc. ACM SIGMOD Conf. on Management of Data*, pp. 145-156.

Korn F, Labrinidis A, Kotidis Y, and Faloutsos C (1998) Ratio rules: a new paradigm for fast, quantifiable data mining. *Proc. 24th Int. Conf. on Very Large Data Base (VLDB)*, 582-593.

Lakshmanan L, Ng R, Han J, and Pang A (1999) Optimization of constrained frequent set queries with 2-variable constraints. *Proc. ACM SIGMOD Conf. on Management of Data*, 157-168.

Lent B, Swami A, and Widom J (1997) Clustering association rules. *Proc. 13th Intl. Conf. on Data Engineering*, 220-231.

Liu B, Hsu W, and Ma Y (1998) Integrating classification and association rule mining. *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 80-86.

Liu B, Hsu W, and Ma Y (1999) Pruning and summarizing discovered associations. *Proc. of ACM SIGKDD.*

Lu H, Setiono R, and Liu H (1996) Effective data mining using neural networks. *Transactions on Knowledge and Data Engineering,* 8:957-961.

Mannila H and Toivonen H (1997) Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1:241-258.

Mannila H, Toivonen H, and Verkamo A (1997) Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259-289.

Ng R, Lakshmanan L, Han J, and Pang A (1998) Exploratory mining and pruning optimizations of constrained association rules. *Proc. ACM SIGMOD Conf. on Management of Data*, pp.13-25.

Rastogi R and Shim K (1998) Mining optimized association rules with categorical and numerical attributes. *Proc. 14th Int. Conf. on Data Engineering (ICDE).*

Rastogi R and Shim K (1999) Mining optimized support rules for numerical attributes. *Proc. 15th Int. Conf. on Data Engineering (ICDE).*

Savasere A, Omiecinski E, and Navathe S (1995) An efficient algorithm for mining association rules in a large databases. *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB).*

Srikant R and Agrawal R (1996) Mining quantitative association rules in large relational tables. *Proc. ACM SIGMOD Conf. on Management of Data.*

Srikant R and Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. *Proc. 5th Int. Conf. on Extending Database Technology (EDBT)*, Avignon, France, March.

Tsur D, Ullman J, Abiteboul S, Clifton C, Motwani R, Nestorov S, Rosenthal A (1998) Query flocks: a generalization of association-rule mining. *Proc. ACM SIGMOD Conf. on Management of Data*, pp. 1-12.

Wang W, Yang J, and Muntz R (1997) STING: a statistical information grid based approach to spatial data mining. *Proc. 23rd Int. Conf. on Very Large Data Base (VLDB)*, pp. 186-195.

Wang K, Tay S, and Liu B (1998) Interestingness-based interval merger for numeric association rules. *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 121-128.

Yang J, Wang W, and Muntz R (2000) Collaborative web caching based on proxy affinities. *Proc. of ACM SIGMETRICS Conference.*

Yip C, Loo K, Kao B, and Cheng C (1999) LGen - a lattice-based candidate set generation algorithm for I/O efficient association rule mining. *Third Pacific-Asia Conference, Lecture Notes in Computer Science,* 1574:54-63.

Yoda K, Fukuda T, Morimoto Y, Morishita S, and Tokuyama T (1997) Computing optimized rectilinear regions for association rules. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining*, pp. 96-103.

Author Biographies

insert photo

Wei Wang currently is an assistant professor in the computer science department at University of North Carolina, Chapel Hill. Her current research interests include data mining, database systems, and bio-informatics. Dr. Wang received the M.S. degree from State University of New York at Binghamton in 1995, and the Ph.D. degree in Computer Science from University of California, Los Angeles in 1999. She is a member of the editorial board of the Journal of Data Management and has published more than forty research papers.

insert photo

Jiong Yang is a visiting assistant professor at computer science department, UIUC. His current research interests include data mining, bio-informatics, Internet traffic engineering, Internet content retrieval, and multimedia systems over the Internet. Dr. Yang received th BS degree from University of California at Berkeley in 1994, the MS and Ph.D. degree in Computer Science from University of California, Los Angeles in 1996 and 1999, respectively. He is the author of more than thirty research papers.

insert photo

Philip S. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is with the IBM Thomas J. Watson Research Center and currently manager of the Software Tools and Techniques group. His research interests include data mining, bioinformatics, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, performance modeling and workload analysis. Dr. Yu has published more than 320 papers in refereed journals and conferences. He holds or has applied for 254 US patents. Dr. Yu is a Fellow of the ACM and a Fellow of the IEEE. He is the Editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering. He is also an associate editor of ACM Transactions on the Internet Technology and that of Knowledge and Information Systems. He is a member of the IEEE Data Engineering steering committee and is also on the steering committee of IEEE Conference on Data Mining. He was an editor and advisory board member of IEEE Transactions on Knowledge and Data Engineering and also a guest co-editor of the special issue on mining of databases. In addition to serving as program committee member on various conferences, he was the program co-chair of the 11th Intl. Conference on Data Engineering and the program chairs of the 2nd Intl. Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the 2nd Intl. Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the general chair of the 14th Intl. Conference on Data Engineering and is currently serving as the general co-chair of the 2nd IEEE Intl. Conference on Data Mining. He has received several IBM and external honors including Best Paper Award, 2 IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, 2 Research Division Awards and the 70th plateau of Invention Achievement Awards. He also received an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. Dr. Yu is an IBM Master Inventor and was recognized as one of the IBM's ten top leading inventors in 1999.

Correspondence and offprint requests to: Wei Wang, Computer Science Department, UNC, Chapel Hill, NC 27510, USA. Email: weiwang@cs.unc.edu