New Algorithms for Fast Discovery of Association Rules *

M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li

Computer Science Department
University of Rochester
Rochester NY 14627
{zaki,srini,ogihara,wei}@cs.rochester.edu

Abstract

Discovery of association rules is an important problem in database mining. In this paper we present new algorithms for fast association mining, which scan the database only once, addressing the open question whether all the rules can be efficiently extracted in a single database pass. The algorithms use novel itemset clustering techniques to approximate the set of potentially maximal frequent itemsets. The algorithms then make use of efficient lattice traversal techniques to generate the frequent itemsets contained in each cluster. We propose two clustering schemes based on equivalence classes and maximal hypergraph cliques, and study two traversal techniques based on bottom-up and hybrid search. We also use a vertical database layout to cluster related transactions together. Experimental results show improvements of over an order of magnitude compared to previous algorithms.

Introduction

One of the central KDD tasks is the discovery of association rules. The prototypical application is the analysis of supermarket sales or *basket* data (Agrawal *et al.* 1996), which can be formally stated as follows: Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be the set of database *items*. Each transaction, T, in the database, $\mathcal{D}$, has a unique identifier, and *contains* a set of items, called an *itemset*. An itemset with k items is called a *k-itemset*. The *support* of an itemset is the percentage of transactions in $\mathcal{D}$ that contain the itemset. An *association rule* is a conditional implication among itemsets, $A \Rightarrow B$. The data mining task for association rules can be broken into two steps. The first step consists of finding all *frequent* itemsets, i.e., itemsets that occur in the database with a certain user-specified frequency, called *minimum support*. The second step consists of forming the rules among the frequent itemsets. This step is relatively easy (Agrawal *et al.* 1996), compared to the computationally intensive first step. Given m items, there are potentially 2^m frequent itemsets, which form a *lattice of subsets* over $\mathcal{I}$. However, only a small fraction of the whole lattice space is frequent. This paper presents efficient methods to discover these frequent itemsets.

Related Work Among the extant solutions, the *Apriori* algorithm (Agrawal *et al.* 1996) was shown to be superior to earlier approaches (Park *et al.* 1995; Holsheimer *et al.* 1995). It uses the *downward closure* property of itemset support to prune the itemset lattice – the property that all subsets of a frequent itemset must themselves be frequent. Thus only the frequent k-itemsets are used to construct *candidate* $(k+1)$-itemsets. A pass over the database is made at each level to find the frequent itemsets. The *Partition* algorithm (Savasere *et al.* 1995) minimizes I/O by scanning the database only twice. Once for generating a set of potential frequent itemsets, and once for gathering their support. Another way to minimize the I/O overhead is to work with only a small sample of the database (Toivonen 1996; Zaki *et al.* 1997a). A number of parallel algorithms have also been proposed (Agrawal & Shafer 1996; Zaki *et al.* 1996; 1997c).

Itemset Clustering

Consider the lattice shown in figure 1. Due to the downward closure property of itemset support the frequent itemsets (dashed circles) form a *border* (bold line), such that all frequent itemsets lie below it. The border is precisely determined by the sub-lattices induced by the maximal frequent itemsets (bold circles). If we are given the maximal frequent itemsets we can design an optimal algorithm that gathers the support of all their subsets in a single database pass. In general we cannot precisely determine the maximal itemsets *a priori*. We can however use intermediate results to obtain their approximations, called the *potential maximal frequent itemsets*.

Equivalence Class Clustering For any $k \geq 2$, we can generate the set of potential maximal itemsets from the set of frequent itemsets, L_k. We partition L_k into

* This work was supported in part by an NSF Research Initiation Award (CCR-9409120) and ARPA contract (F19628-94-C-0057).

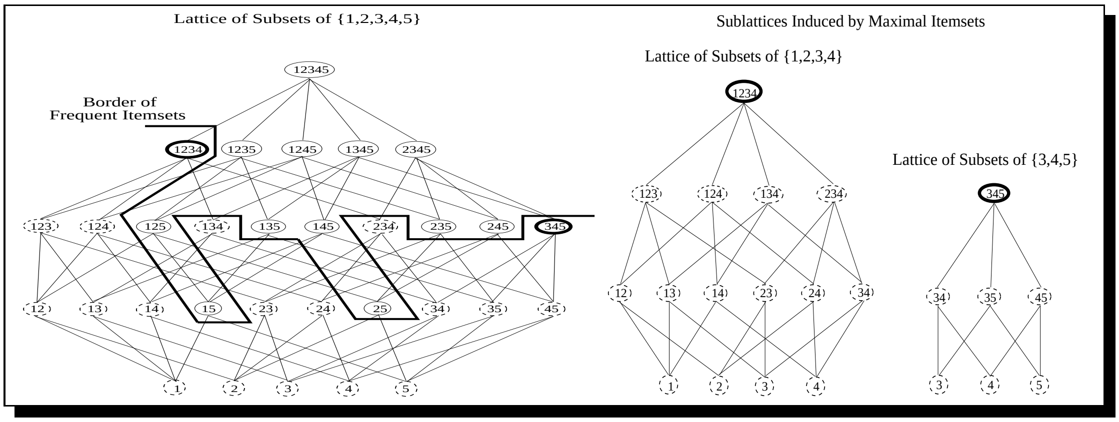


Figure 1: Lattice of Subsets and Maximal Itemset Induced Sub-lattices

equivalence classes based on their common $k-1$ length prefix, given as, $[\mathbf{a}] = \{b[k] | a[1:k-1] = b[1:k-1]\}$. For example, consider the L_2 and the resulting equivalence classes shown in figure 2. Any frequent itemset with the prefix 1, must consist of items in $[\mathbf{1}]$, making 12345678 a potential maximal itemset. Each equivalence class can thus be considered as a potential maximal frequent itemset. Note that for $k = 1$ we end up with the entire item universe as the maximal itemset. However, for any $k \geq 2$, we can extract more precise knowledge, with increasing precision as k increases.

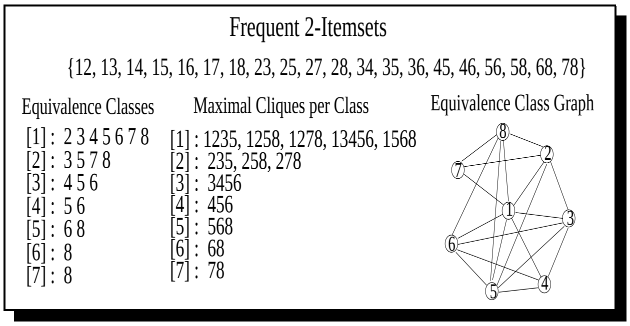


Figure 2: Clustering Schemes

Maximal Hypergraph Clique Clustering From L_k, it is possible to generate a more refined set of potential maximal itemsets. The key observation is that given any frequent m-itemset, for $m > k$, all its k-subsets must be frequent. In graph-theoretic terms, if each item is a vertex in a hypergraph, and each k-subset an edge, then the frequent m-itemset must form a k-uniform hypergraph clique. Furthermore, the set of maximal hypergraph cliques represents an approximation or upper-bound on the set of maximal potential frequent itemsets. All the *true* maximal frequent itemsets are contained in the vertex set of the maximal cliques, as stated formally in the lemma below.

Lemma 1 *Let H_{L_k} be the k-uniform hypergraph with vertex set $\mathcal{I}$, and edge set L_k. Let C be the set of maximal hypergraph cliques in H, and let M be the set of vertex sets of the cliques in C. Then for all maximal frequent itemsets f, $\exists t \in M$, such that $f \subseteq t$.*

An example of maximal hypergraph clique clustering is given in figure 2. The figure shows all the equivalence classes, the maximal cliques per class, and the hypergraph for class $[\mathbf{1}]$. It can be seen immediately that clique clustering is more precise than equivalence class clustering. For example, for the class $[\mathbf{1}]$, the former generated the maximal element 12345678, while the latter a more refined set $\{1235, 1258, 1278, 13456, 1568\}$. The maximal cliques are discovered using a dynamic programming algorithm; see (Zaki *et al.* 1997b) for details. As the edge density of the equivalence class graph increases the cost for generating the cliques may increase. Some of the factors affecting the edge density include decreasing support and increasing transaction size.

Lattice Traversal

Each potential maximal itemset generated by the above clustering schemes, induces a sublattice on $\mathcal{I}$. We now have to traverse each of these sub-lattices to determine the true frequent itemsets.

Bottom-up Traversal A pure bottom-up lattice traversal proceeds in a breadth-first manner generating all frequent itemsets of length k, before generating those of length $k+1$. Figure 3 shows an example of this scheme, with the potential maximal itemset, 123456, and the true maximal frequent itemsets 1235 and 13456. Most current algorithms use this approach (Agrawal *et al.* 1996; Savasere *et al.* 1995; Park *et al.* 1995).

Hybrid Top-down/Bottom-up Traversal The bottom-up search may generate spurious candidates in the intermediate steps, since the fact that all subsets of an itemset are frequent doesn't guarantee that the itemset is frequent. We can envision other traversal techniques which quickly identify the set of true maximal frequent itemsets. If we are interested in all frequent itemsets, we can then gather the support of all their subsets as well. We rule out a pure top-down ap-

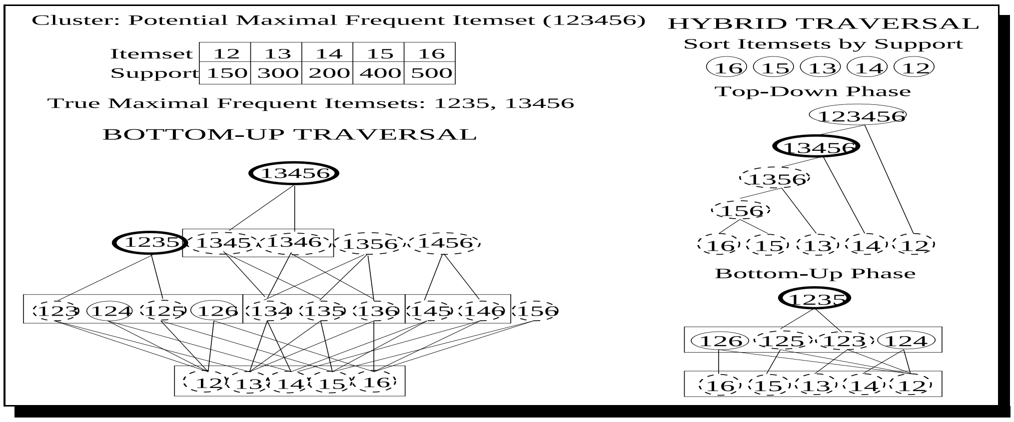


Figure 3: Bottom-up and Hybrid Lattice Traversal

proach due to the inaccuracies in the clusters (Zaki *et al.* 1997b), and propose a hybrid top-down/bottom-up scheme that works well in practice. The basic idea is to start with a single element from the itemset cluster, and extend this by one more element till we generate an infrequent itemset. This comprises the top-down phase. In the bottom-up phase, the remaining elements are combined with the elements in the first set to generate all the additional frequent itemsets. For the top-down phase, we sort the cluster elements in descending order of their support. We start with the element with maximum support, and extend it with the next element in the sorted order. This is based on the intuition that the larger the support the more likely is the itemset to be part of a larger itemset. Figure 3 shows an example of the hybrid scheme.

Transaction Clustering

There are two possible layouts of the database for association mining. The *horizontal* layout (Agrawal *et al.* 1996) consists of a list of transactions. Each transaction has an identifier followed by a list of items. The *vertical* layout (Holsheimer *et al.* 1995) consists of a list of items. Each item has a *tid-list* – the list of all the transactions containing the item. The vertical format seems more suitable for association mining since the support of a candidate k-itemset can be computed by simple tid-list intersections. No complicated data structures need to be maintained. The tid-lists cluster relevant transactions, and avoid scanning the whole database to compute support, and the larger the itemset, the shorter the tid-lists, resulting in faster intersections. Furthermore, the horizontal layout seems suitable only for the bottom-up traversal. The inverted layout, however, has a drawback. Intersecting 1-itemset tid-lists to determine L_2 can be very expensive (Zaki *et al.* 1997b). This can be solved by using sampling(Toivonen 1996; Zaki *et al.* 1997a), or by using a preprocessing step to gather the support all 2-itemsets. Since this information is invariant, the pre-processing has to be performed once initially, and the cost can be amortized over the number of times the data is mined. Our current implementation uses the pre-processing approach due to its simplicity. Sampling requires an extra database pass, while pre-processing requires extra storage. For m items, $\mathcal{O}(m^2)$ disk space is required, which can be quite large for large m. However, for $m = 1000$ used in our experiments this adds only a very small extra storage overhead. Note that the database itself requires the same amount of space in both the horizontal and vertical formats.

New Association Algorithms

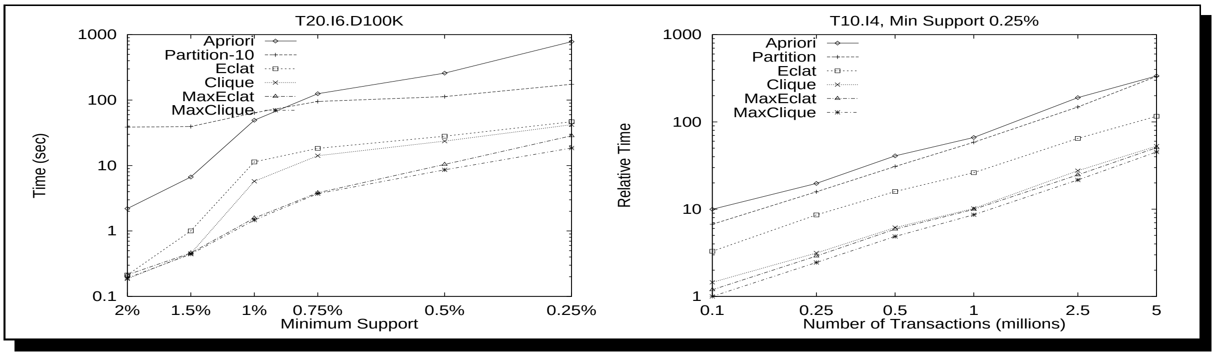
We present four new algorithms, depending on the clustering and lattice traversal scheme used:

- *Eclat*: equivalence class & bottom-up
- *MaxEclat*: equivalence class & hybrid
- *Clique*: maximal hypergraph clique & bottom-up
- *MaxClique*: maximal hypergraph clique & hybrid

The new algorithms use one of the itemset clustering schemes to generate potential maximal itemsets. Each such cluster induces a sublattice, which is traversed using bottom-up search to generate all frequent itemsets, or using hybrid scheme to generate only the maximal frequent itemsets. Each cluster is processed in its entirety before moving on to the next cluster. Since the transactions are clustered using the vertical format, this involves a single database scan, resulting in huge I/O savings. Frequent itemsets are determined using simple tid-list intersections. No complex hash structures need to be built or searched. The algorithms have low memory utilization, since only the frequent k-itemsets within a single cluster need be kept in memory at any point. The use of simple intersection operations also makes the new algorithms an attractive option for direct implementation on general purpose database systems.

Experimental Results

Our experiments used a 100MHz MIPS processor with 256MB main memory, with different benchmark databases (Agrawal *et al.* 1996). For fair comparison, all algorithms use 2-itemset supports from the preprocessing step. See (Zaki *et al.* 1997b) for detailed ex-



(a) Total Execution Time (b) Number of Transactions Scale-up

Figure 4: Performance Comparison

periments. In figure 4 a), we compare our new algorithms against *Apriori* and *Partition* (with 10 partitions) on T20.I6.D100K database. *Eclat* outperforms *Apriori* by a factor of 10, and *Partition* by a factor of 5. As the support decreases, the size and the number of frequent itemsets increases. *Apriori* has to make multiple passes over the database, and performs poorly. *Partition* saves some I/O costs, but it spends time computing redundant frequent itemsets in common among different partitions. Among the new algorithms, *Clique* provides a finer level of clustering, reducing the number of candidates considered, and performs better than *Eclat*. Both the hybrid algorithms, *MaxEclat* and *MaxClique*, outperform the bottom-up ones, since they only find maximal itemsets, and thus perform fewer joins. Table 1 gives the number of joins performed by the different algorithms. Compared to *Eclat*, the hypergraph clique clustering is able to cut down the joins by 25% for *Clique*. Combined with the hybrid search, there is a 75% reduction for *MaxClique*, making it the best algorithm. It outperforms *Apriori* by a factor of 40, *Partition* by a factor of 20, and *Eclat* by a factor of 2.5.

	Eclat	Clique	MaxEclat	MaxClique
# Joins	83606	61968	56908	20322
Time (sec)	46.7	42.1	28.5	18.5

Table 1: Number of Joins: T20.I6.D100K (0.25%)

Figure 4 b) shows how the different algorithms scale up as the number of transactions increases from 0.1 to 5 million (M). The times are normalized against the execution time for *MaxClique* on 0.1M transactions. The number of partitions for *Partition* was varied from 1 to 50. While all the algorithms scale linearly, the slope is much smaller for the new algorithms. The new algorithms also scale well with transaction size, and have very low memory utilization (Zaki *et al.* 1997b).

Conclusions

We proposed new algorithms for fast association mining, using three main techniques. We first cluster itemsets using equivalence classes or maximal hypergraph cliques. We then generate the frequent itemsets from each cluster using bottom-up or hybrid traversal. A vertical database layout is used to cluster transactions, enabling us to make only one database scan. Experimental results indicate more than an order of magnitude improvements over previous algorithms.

References

Agrawal, R. & Shafer, J. 1996. Parallel mining of association rules. In *IEEE Knowledge & Data Engg.*, 8(6):962–969.

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; & Verkamo, A. 1996. Fast discovery of association rules. In *Advances in KDD.* MIT Press.

Holsheimer, M.; Kersten, M.; Mannila, H.; & Toivonen, H. 1995. A perspective on databases and data mining. In *1st KDD Conf.*

Park, J.; Chen, M.; & Yu, P. 1995. An effective hash based algorithm for mining association rules. In *SIGMOD Conf.*

Savasere, A.; Omiecinski, E.; and Navathe, S. 1995. An efficient algorithm for mining association rules in large databases. In *21st VLDB Conf*

Toivonen, H. 1996. Sampling large databases for association rules. In *22nd VLDB Conf*

Zaki, M.; Ogihara, M.; Parthasarathy, S.; & Li, W. 1996. Parallel data mining for association rules on shared-memory multi-processors. In *Supercomputing.*

Zaki, M.; Parthasarathy, S.; Li, W.; & Ogihara, M. 1997a. Evaluation of sampling for data mining of association rules. In *7th Wkshp. Resrch. Iss. Data Engg.*

Zaki, M.; Parthasarathy, S.; Ogihara, M.; & Li, W. 1997b. New algorithms for fast discovery of association rules. TR 651, CS Dept, Univ. of Rochester.

Zaki, M.; Parthasarathy, S.; & Li, W. 1997c. A localized algorithm for parallel association mining. In *9th ACM Symp. Parallel Algorithms & Architectures.*