Mining Large Itemsets for Association Rules

Charu C. Aggarwal and Philip S. Yu
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

This paper provides a survey of the itemset method for association rule generation. The paper discusses past research on the topic and also studies the relevance and importance of the itemset method in generating association rules. We discuss a number of variations of the association rule problem which have been proposed in the literature and their practical applications. Some inherent weaknesses of the large itemset method for association rule generation have been explored. We also discuss some other formulations of associations which can be viable alternatives to the traditional association rule generation method.

1 Introduction

Association rules find the relationships between the different items in a database of sales transactions. Such rules track the buying patterns in consumer behavior eg. finding how the presence of one item in the transaction affects the presence of another and so forth. The problem of association rule generation has recently gained considerable prominence in the data mining community because of the capability of its being used as an important tool for knowledge discovery. Consequently, there has been a spurt of research activity in the recent years surrounding this problem.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of binary literals called items. Each transaction T is a set of items, such that $T \subseteq I$. This corresponds to the set of items which a consumer may buy in a basket transaction.

An *association rule* is a condition of the form $X \Rightarrow Y$ where $X \subseteq I$ and $Y \subseteq I$ are two sets of items. The idea of an association rule is to develop a systematic method by which a user can figure out how to infer the presence of some sets of items, given the presence of other items in a transaction. Such information is useful in making decisions such as customer targeting, shelving, and sales promotions.

An important approach to the association rule problem was developed by Agrawal et. al. in [4]. This is a two-phase large itemset approach. Certain terminologies were defined in the same work which formalize these notions. In order to discuss the method further, we shall review some important definitions.

The *support* of a rule $X \Rightarrow Y$ is the fraction of transactions which contain both X and Y.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

The *confidence* of a rule $X \Rightarrow Y$ is the fraction of transactions containing X, which also contain Y. Thus, if we say that a rule has 90% confidence then it means that 90% of the tuples containing X also contain Y.

The large itemset approach is as follows. Generate all combinations of items that have fractional transaction support above a certain user-defined threshold called *minsupport*. We call all such combinations *large itemsets*. Given an itemset $S = \{i_1, i_2, \ldots, i_k\}$, we can use it to generate at most k rules of the type $[S - \{i_r\}] \Rightarrow \{i_r\}$ for each $r \in \{1, \ldots, k\}$. Once these rules have been generated, only those rules above a certain user defined threshold called *minconfidence* may be retained.

In order to generate the large itemsets, an iterative approach is used to first generate the set of large 1-itemsets L_1, then the set of large itemsets L_2, and so on until for some value of r the set L_r is empty. At this stage the algorithm can be terminated. During the kth iteration of this procedure a set of candidates C_k is generated by performing a $(k-2)$-join on the large itemsets L_{k-1}. The itemsets in this set C_k are *candidates* for large itemsets, and the final set of large itemsets L_k must be a subset of C_k. Each element of C_k needs to be validated against the transaction database to see if it indeed belongs to L_k. The validation of the candidate itemset C_k against the transaction database seems to be bottleneck operation for the algorithm. This method requires multiple passes over a transaction database which may potentially be quite large. For evaluating itemsets with a specific number of items, one pass is required over the transaction database. Thus, if the large itemset with the maximum number of items has 9 items in it, then the method requires 9 passes over the transaction database. This may result in substantial I/O times for the algorithm.

2 A survey of research on the large itemset method

After the inital algorithms proposed by Agrawal et. al. in [4], the problem has been extensively studied by other researchers and a number of fast variants have been proposed. In a subsequent paper in [5], Agrawal et. al. has discussed how the the algorithm for finding large itemsets may be sped up substantially by introducing a pruning approach which reduces the size of the candidate C_k. This algorithm uses the pruning trick that all subsets of a large itemset must also be large. Thus, if some $(k-1)$-subset of an itemset $I \in C_k$ does not belong to L_{k-1}, then that itemset can be pruned from further consideration. This process of pruning eliminates the need for finding the support of the candidate itemset I. In the same paper [5], an efficient data structure known as the hash-tree was introduced for evaluating the support of an itemset.

Subsequent work on the large itemset method has concentrated on the following aspects:

(1) Improving the I/O costs by reducing the number of passes over the transaction database.

(2) Improving the computational efficiency of the large itemset generation procedure.

(3) Finding efficient parallel algorithms for association rule generation.

(4) Introducing sampling techniques for improving the I/O and computational costs of large itemset generation.

(5) Extensions of the large itemset method to other problems such as quantitative association rules, generalized associations, and cyclic association rules.

(6) Finding methods for online generation of association rules by using the preprocess-once-query-many paradigm of online analytical processing.

2.1 Alternative Enhancements

We shall provide a brief survey of the work done in each of the above categories. A hash-based algorithm for efficiently finding large itemsets was proposed by Park et. al. in [19]. It was observed that most of the time in the was spent in evaluating and finding large 2-itemsets. The algorithm in Park et. al.[19] attempts to improve this approach by providing a hash based algorithm for quickly finding large 2-itemsets.

Brin et. al. proposed a method for large itemset generation which reduces the number of passes over the transaction database by counting some $(k+1)$-itemsets in parallel with counting k-itemsets. In most previously proposed algorithms for finding large itemsets, the support for a $(k+1)$-itemset was counted after k-itemsets have already been generated. In this work, it was proposed that one could start counting a $(k+1)$-itemset as soon as it was suspected that this itemset might be large. Thus, the algorithm could start counting for $(k+1)$-itemsets much earlier than completing the counting of k-itemsets. The total number of passes required by this algorithm is usually much smaller than the maximum size of a large itemset.

A partitioning algorithm was proposed by Savasere et. al. [21] for finding large itemsets by dividing the database into n partitions. The size of each partition is such that the set of transactions can be maintained in main memory. Then, large itemsets are generated separately for each partition. Let LP_i be the set of large itemsets associated with the ith partition. Then, if an itemset is large, then it must be the case that it must belong to at least one of LP_i for $i \in \{1, \ldots, k\}$. Now, the support of the candidates $\cup_{i=1}^{k} LP_i$ can be counted in order to find the large itemsets. This method requires just two passes over the transaction database in order to find the large itemsets.

The approach described above is highly parallelizable, and has been used to generate large itemsets by assigning each partition to a processor. At the end of each iteration of the large itemset method the processors need to communicate with one another in order to find the global counts of the candidate k-itemsets. Often, this communication process may impose a substantial bottleneck on the running time of the algorithm. In other cases, the time taken by the individual processors in order to generate the processor-specific large itemsets may be the bottleneck. Other methods have been proposed such as using a shared hash-tree among the multi-processors in order to generate the large itemsets[28]. More work on the parallelization of the itemset method may be found in [7, 17, 20, 28, 29].

A common feature of most of the algorithms reviewed above and proposed in the literature is that most such research is are variations on the “bottom-up theme” proposed by the *Apriori* algorithm[4, 5]. For databases in which the itemsets may be long, these algorithms may require substantial computational effort. Consider for example a database in which the length of the longest itemset is 40. In this case, there are 2^{40} subsets of this single itemset, each of which would need to be validated against the transaction database. Thus, the success of the above algorithms critically relies on the fact that the length of the frequent patterns in the database are typically short. An interesting algorithm for itemset generation has been proposed very recently by Bayardo [8]. This algorithm uses clever “look-ahead” techniques in order to identify longer patterns earlier on. The subsets of these patterns can then be pruned from further consideration. Initial computational results [8] indicate that the algorithm can lead to substantial performance improvements over the *Apriori* method.

Since the size of the transaction database is typically very large, it may often be desirable to use random sampling in order to generate the large itemsets. The use of random sampling to generate large itemsets may save considerable expense in terms of the I/O costs. A method of random sampling was introduced by Toivonen in [27]. The weakness of using random sampling is that it may often result in inaccuracies because of the presence of *data skew*. Data which are located on the same page may often be highly correlated and may not represent the over all distribution of patterns through the entire database. As a result, it may often be the case that sampling just 5% of the transactions may be as expensive as a pass through the entire database. Anti-skew algorithms for mining association rules have been discussed by Lin and Dunham in [16]. The techniques proposed in this paper reduce the maximum number of scans required to less than 2. The algorithms use a sampling process in order to collect knowledge about the data and reduce the number of passes.

The problems created by data skewness also arise in the context of parallel methods which divide the load among processors by partitioning the transaction data among the different processors. This is because each processor may receive a set of transactions which are not very representative of the entire data set. The problems created by skewness of the data in parallel mining of association rules have been investigated in [25].

2.2 Generalizations of the association rule problem

Initially, the association rule problem was proposed in the context of supermarket data. The motivation was to find how the items bought in a consumer basket related to each other. A number of interesting extensions and applications have been proposed. The problem of mining quantitative association rules in relational tables was proposed in [26]. In such cases association rules are discovered in relational tables which have both categorical and quantitative attributes. Thus, it is possible to find rules which indicate how a given range of quantitative and categorical attributes may affect the values of other attributes in the data. The algorithm for the quantitative association rule problem discretizes the quantitative data into disjoint ranges and then constructs an item corresponding to each such range. Once these pseudo-items have been constructed, a large itemset procedure can be applied in order to find the association rules. Often a large number of rules may be produced by such partitioning methods, many of which may not be interesting. An interest measure was defined and used in [26] in order to generate the association rules. In [14], an algorithm for clustering quantitative association rules was proposed. The aim of this algorithm was to generate rules which were more natural in terms of the quantitative clusters with which individual rules were associated. A closely related issue to finding quantitative association rules is the problem of finding profile association rules in which it is desirable to tie together rules which tie together user profiles with buying patterns. An algorithm for finding profile association rules was discussed in [2]. A method for finding optimized quantitative association rules has been discussed in [26]. This paper discusses how to choose the quantitative ranges in an optimal way so as to maximize the strength of the given association rules.

An interesting issue is that of handling taxonomies of items. For example, in a store, there may be several kinds of cereal, and for each individual kind of cereal, there may be multiple brands. Rules which handle such taxonomies are called *generalized associations*. The motivation is to generate rules which are as general as possible and also as general as possible while taking such taxonomies into account. Algorithms for finding such rules were presented in [24]. Savasere et. al.[22] also discuss how to find interesting negative association rules in the context of taxonomies of items. The focus of this work is to find rules which negatively correlate with rules which are discovered at higher levels of the taxonomy. Another useful extension of association rules which has been recently been proposed is the concept of *cyclic association rules*. It may often be the case that when association rules are computed for data which have a time component, periodic seasonal variations may be observed. For example, the monthly sales of goods correlate with each other differently on a seasonal basis. Ozden et. al. [18] have proposed efficient algorithms for finding association rules which display cyclic variation over time.

2.3 Online generation of association rules

Since the size of the transaction database may be very large, the algorithms for finding association rules are both compute-intensive and require substantial I/O. Thus it is difficult to provide quick responses to user queries. Methods for online generation of association rules have been discussed by Aggarwal and Yu[1]. This algorithm uses the preprocess-once-query-many paradigm of OLAP in order to generate association rules quickly by using an adjacency lattice to prestore itemsets. The interesting feature of this work is that the rules which are generated are independent of both the size of the transaction data and the number of itemsets prestored. In fact, the running time of the algorithm is completely proportional to the size of the output. It is also possible to generate queries for rules with specific items in them. In the same work, redundancy in association rule generation has been discussed. A rule is said to be redundant at a given level of support and confidence if its existance is implied by

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Rule	**Support**	**Confidence**
$X \Rightarrow Y$	25%	50%
$X \Rightarrow Z$	37.5%	75%

Table 1: (a) The base data (b) Corresponding support and confidence

some other rule in the set. For example, consider the following pair of rules:

$$\{\text{Milk}\} \Rightarrow \{\text{Bread, Butter}\}$$
$$\{\text{Milk, Bread}\} \Rightarrow \{\text{Butter}\}$$

In the above example, the second rule is redundant since its existance is implied by the first. Typically, a single essential rule may imply a large number of rules in the final output. Algorithms were proposed to generate a minimal set of essential rules for a given set of data.

3 A criticism of the large itemset method

The large itemset method has proved as a useful tool for mining association rules in large datasets which are inherently sparse. However, in many cases the method is difficult to generalize to other scenarios for computational reasons. Some of the criticisms associated with the large itemset method are the following:

- **Spuriousness in itemset generation:** We shall explain this issue with the help of an example. Consider a retailer of breakfast cereal which surveys 5000 students on the activities they engage in the morning. The data shows that 3000 students play basketball, 3750 eat cereal, and 2000 students both play basketball and eat cereal. For a minimum support of 40%, and minimum confidence of 60%, we find the following association rule:
  play basketball ⇒ eat cereal
  The association rule is misleading because the overall percentage of students eating cereal is 75% which is even larger than 60%. Thus, playing basketball and eating cereals are negatively associated: being involved in one decreases the chances of being involved in the other. Consider the following association:
  play basketball ⇒ (not) eat cereal
  Although this rule has both lower support and lower confidence than the rule implying positive association, it is far more accurate. Thus, if we set the support and confidence sufficiently low, two contradictory rules would be generated; on the other hand if we set the parameters sufficiently high only the inaccurate rule would be generated. In other words, no combination of support and confidence can generate purely the correct association.

  Another example is illustrated in Table 1(a), in which we have three items X, Y, and Z. The coefficient of correlation between X and Y is 0.577, while that between X and Z is -0.378. Thus, X and Y are positively related to each other, while X and Z are negatively related. The support for the rules $X \Rightarrow Y$ and $X \Rightarrow Z$ are illustrated in Table 1(b). Interestingly, the support and confidence for the rule $X \Rightarrow Z$ strictly dominates the support and confidence for the rule $X \Rightarrow Y$. Thus, it is not possible to find any level of *minsupport* and *minconfidence* at which only the rule $X \Rightarrow Y$ would be generated, without a spurious rule $X \Rightarrow Z$ being generated as well. If we set *minsupport* and *minconfidence* too low, then in many cases (especially when different items have widely varying global density) an unacceptably large number of rules might be generated, (a great majority of which may be spurious) and this defeats the purpose of data mining in the first place.

- **Dealing with dense data sets:** For a k-dimensional database, there are 2^k possibilities for itemsets. Some data sets may be such that a large number of these 2^k possibilities may qualify above the minimum support. For such situations, it may be necessary to set the *minsupport* to an unacceptably high level. This may result in a number of important rules being lost. Often, the value of *minsupport* is decided based on computational constraints or in restricting the number of rules generated to a manageable number. Thus, it is impossible to ascertain when important rules are being lost and when they are not.

  Some important applications in which the data may be dense are the following:

  - **Negative association rules:** Suppose we wish to find negative correlations among items. In other words, we wish to mine association rules in which presence as well as absence of an item may be used. Although the large itemset approach can be directly extended to this problem by treating the absence of an item as a pseudo-item, the sparsity of item presence in real transaction data may result in considerable bias towards rules which are concerned only with finding rules corresponding to absence of items rather than their presence.
  - **Data in which different attributes have widely varying densities:** Suppose that the categorical data corresponding to customer profile information is available along with the sales transaction data and we wish to find association rules concerning the demographic nature of people buying different items. In this case, directly applying the itemset method may be very difficult because of the different densities of the demographic and the sales transaction information. An example of such a demographic attribute is sex, which may take on either value 50% of the time. On the other hand a bit representing an item may take on the value of 1 only 5% of the time.

4 An alternative approach using collective strength

The use of an interest measure has been presented as a solution in order to avoid spurious association rules. The interest level of a rule is the ratio of the actual strength to the expected strength based upon the assumption of statistical independence. Past work has concentrated on using the interest measure as a pruning tool in order to remove the uninteresting rules in the output. However, (as the basketball-cereal example illustrates) as long as support is still the primary determining factor in the initial itemset generation, either the user has to set the initial support parameter low enough so as to not lose any interesting rules in the output or risk losing some important rules. In the former case, computational efficiency may be a problem, while the latter case has the problem of not being able to retain rules which may be interesting from the point of view of a user.

There is some other work which deals with providing alternative methods for viewing itemsets. Correlation rules have been discussed by Brin et. al. in [9]. In [10], the notion of developing implication rules instead of association rules has been discussed. The implication strength of a rule is a number between 0 and ∞. An implication strength of 1 indicates that the strength of the rule is exactly as it would be under the assumption of statistical independence. An implication strength which is larger than 1 indicates greater than expected presence. This measure is preferable to confidence because it deals with greater than expected measures for finding association rules.

A different notion called “collective strength” has been defined in [3]. In this work, it is desired to find interesting association rules by using “greater than expected values”.

The collective strength of an itemset is defined to be a number between 0 to ∞. A value of 0 indicates perfect negative correlation, while a value of ∞ indicates perfectly positive correlation. A value of 1 indicates the “break-even point”, corresponding to an itemset present at expected value.

An itemset I is said to be in *violation* of a transaction, if some of the items are present in the transaction, and others are not. Thus, the concept of violation denotes how many times a customer may buy at least some of the items in the itemset, but may not buy the rest of the items.

The *violation rate* of an itemset I is denoted by $v(I)$ and is the fraction of violations of the itemset I over all transactions.

The *collective strength* of an itemset I is denoted by $C(I)$ and is defined as follows:

$$C(I) = \frac{1 - v(I)}{1 - E[v(I)]} \cdot \frac{E[v(I)]}{v(I)} \tag{1}$$

The expected value of of $v(I)$ is calculated assuming statistical independence. Let us pause here to understand the meaning of collective strength before proceeding further. We note that the violation of an itemset in a transaction is a "bad event" from the perspective of trying to establish high correlation among the corresponding items. Thus $v(I)$ is the fraction of bad events while $(1 - v(I))$ is the fraction of "good events". The above definition for collective strength can be recast as follows:

$$C(I) = \frac{\text{Good Events}}{\text{E[Good Events]}} \cdot \frac{\text{E[Bad Events]}}{\text{Bad Events}} \tag{2}$$

The following are some of the interesting properties of collective strength.

(1) The notion of collective strength treats the 0 and 1 attributes in a symmetric way. Thus, if we were to apply this to a problem in which the absence as well as the presence of an item is used to find the itemsets, then we can immediately perceive the benefits of this definition.

(2) It is instructive to examine how this definition for collective strength fares for the case of a 2-itemset $I = \{i_1, i_2\}$. If i_1 and i_2 items are perfectly positively correlated, then the collective strength for the corresponding 2-itemset is ∞. This is because in this case, the fraction of occurances of bad events is 0. On the other hand, when the items are perfectly negatively correlated, the fraction of good events is 0, and hence the collective strength for the corresponding itemset pair is 0. For items which are independent from one another, the collective strength is 1. We provide this example in light of the criticisms that we expounded for the use of the interest measure at the beginning of the section.

(3) The collective strength uses the *relative* number of times an itemset is present in the database. The itemsets which have an insignificant presence can always be pruned off at a later stage. The level of presence of an itemset I which constitutes useful information is not exactly the same problem as finding whether or not the items in I are related to each other. Support can still be used in order to isolate those itemsets which have substantial presence in the database.

A *strongly collective itemset* is one which has high collective strength along with all of its subsets.

Definition 1: An itemset I is denoted to be strongly collective at level K, if it satisfies the following properties:

(1) The collective strength $C(I)$ of the itemset I is at least K.

(2) **Closure property:** The collective strength $C(J)$ of every subset J of I is at least K.

It is necessary to force the closure property in order to ensure that unrelated items may not be present in a itemset. Consider, for example, the case when itemset I_1 is $\{$Milk, Bread$\}$ and itemset I_2 is $\{$Diaper, Beer$\}$. If I_1 and I_2 each have high collective strength, then it may often be the case that the itemset $I_1 \cup I_2$ may also have a high collective strength, even though items such as milk and beer may be independent.

Algorithms for finding strongly collective itemsets have been proposed in [3]. This algorithm requires at most 2 passes over the transaction database in order to find the strongly collective baskets.

5 Conclusions and Summary

This paper discusses a survey of the large itemset method and its applications. The amount of research devoted to this problem has been very substantial in the recent years. Increasingly efficient algorithms have been proposed for the large itemset method by using smart modifications on the *Apriori* algorithm proposed by Agrawal et. al. However, the considerable computational requirements of the *Apriori* method may often require methods which are capable of online generation of association rules. Such techniques have been discussed by Aggarwal and Yu in [1]. We also discussed the weaknesses of the large itemset method for association rule generation. An alternative to the previously proposed large itemset technique has been discussed in [3] which uses greater than expected measures in order to generate association rules. This technique may be often preferable to to the large itemset method when the data is either fully or partially dense.

References

[1] Aggarwal C. C., and Yu P. S. “Online Generation of Association Rules.” *Proceedings of the International Conference on Data Engineering,* Orlando, Florida, February 1998.

[2] Aggarwal C. C., Sun Z., and Yu P. S. “Generating Profile Association Rules.” *IBM Research Report*, RC-21037.

[3] Aggarwal C. C., and Yu P. S. “A new framework for itemset generation.” *IBM Research Report*, RC-21064.

[4] Agrawal R., Imielinski T., and Swami A. “Mining association rules between sets of items in very large databases.” *Proceedings of the ACM SIGMOD Conference on Management of data,* pages 207-216, 1993.

[5] Agrawal R., and Srikant R. “Fast Algorithms for Mining Association Rules in Large Databases.” *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 478-499, 1994.

[6] Agrawal R., and Srikant R. “ Mining Sequential Patterns.” *Proceedings of the 11th International Conference on Data Engineering,* pages 3-14, March 1995.

[7] Agrawal R. and Shafer J. “Parallel Mining of Association Rules: Design, Implementation, and Experience.” *Technical Report RJ10004, IBM Almaden Research Center*, San Jose, CA 95120, Jan. 1996.

[8] Bayardo R. J. “Efficiently Mining Long Patterns from Databases.” *Unpublished Research Report*.

[9] Brin S., Motwani R. and Silverstein C. “Beyond Market Baskets: Generalizing Association Rules to Correlations.” *Proceedings of the ACM SIGMOD, 1997.* pages 265-276.

[10] Brin S., Motwani R. Ullman J. D. and Tsur S. “Dynamic Itemset Counting and implication rules for Market Basket Data.” *Proceedings of the ACM SIGMOD, 1997.* pages 255-264.

[11] Chen M. S., Han J., and Yu P. S. “Data Mining: An Overview from Database Perspective.” *IEEE Transactions on Knowledge and Data Engineering,* 8(6): 866-883, December 1996.

[12] Klementtinen M., Mannila H., Ronkainen P., Toivonen H., and Verkamo A. I. “Finding interesting rules from large sets of discovered association rules.” *Proceedings of the CIKM* 1994.

[13] Han J., and Fu Y. “Discovery of Multi-level Association Rules From Large Databases.” *Proceedings of the International Conference on Very Large Databases,* pages 420-431, Zurich, Switzerland, September 1995.

[14] Lent B., Swami A., and Widom J. “Clustering Association Rules.” *Proceedings of the Thirteenth International Conference on Data Engineering.* pages 220-231, Birmingham, UK, April 1997.

[15] Mannila H., Toivonen H., and Verkamo A. I. "Efficient algorithms for discovering association rules." *AAAI Workshop on Knowledge Discovery in Databases,* 1994, pages 181-192.

[16] Lin J.-L. and Dunham M. H. "Mining Association Rules: Anti-Skew Algorithms." *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, February 1998.

[17] Mueller A. "Fast sequential and parallel methods for association rule mining: A comparison." Technical Report CS-TR-3515, Department of Computer Science, University of Maryland, College Park, MD, 1995.

[18] Ozden B., Ramaswamy S, and Silberschatz A. "Cyclic Association Rules." *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, February 1998.

[19] Park J. S., Chen M. S., and Yu P. S. "An Effective Hash-based Algorithm for Mining Association Rules." *Proceedings of the ACM-SIGMOD Conference on Management of Data, 1995.* Extended version appears as: "Using a Hash-based Method with Transaction Trimming for Mining Association Rules." *IEEE Transactions ob Knowledge and Data Engineering,* Volume 9, no 5, September 1997, pages 813-825.

[20] Park J. S., Chen M. S. and Yu P. S. "Efficient Parallel Data Mining of Association Rules." *Fourth International Conference on Information and Knowledge Management*, Baltimore, Maryland, November 1995, pages 31-36. Technical Report RC20156, IBM T. J. Watson Research Center, August 1995.

[21] Savasere A., Omiecinski E., and Navathe S. B. "An efficient algorithm for mining association rules in large databases." *Proceedings of the 21st International Conference on Very Large Databases,* 1995.

[22] Savasere A., Omiecinski E., and Navathe S. B. "Mining for Strong Negative Associations in a Large Database of Customer Transactions." *Proceedings of the International Conference on Data Engineering,* February 1998.

[23] Rastogi R., and Shim K. "Mining Optimized Association Rules with Categorical and Numeric Attributes." *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, February 1998.

[24] Srikant R., and Agrawal R. "Mining Generalized Association Rules." *Proceedings of the 21st International Conference on Very Large Data Bases,*1995, pages 407-419.

[25] Xiao Y. and Cheung D. W. "Effects of data Skewness and Workload Balance in Parallel Data Mining." *Unpublished Research Report.*

[26] Srikant R., and Agrawal R. "Mining quantitative association rules in large relational tables". *Proceedings of the ACM SIGMOD Conference on Management of Data, 1996.* pages 1-12.

[27] Toivonen H. "Sampling Large Databases for Association Rules". *Proceedings of the 22nd International Conference on Very Large Databases*, Bombay, India, September 1996.

[28] Zaki M. J., Ogihara M., Parthasarathy S., and Li W. "Parallel Data Mining for Association Rules on Shared-Memory Multi-processors." *Supercomputing'96*, Pittsburg, PA, pages 17-22, 1996 (also available as URCS Technical Report 618 , May 1996).

[29] Zaki M. J., Parthasarathy S., Li W., "A Localized Algorithm for Parallel Association Mining", *9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp 321-330, Newport, Rhode Island, June 22-25, 1997.