Approximate Frequent Itemsets Compression Using Dynamic Clustering Method

Hua Yan, Yongsheng Sang
Computational Intelligence Laboratory
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, P.R China
{huayan, sangys}@uestc.edu.cn

***Abstract*—Frequent-itemsets mining often faces the problem of generating a large collection of frequent itemsets, which is too large to be carefully examined and understood by the users. To reduce the output size of frequent itemsets, we propose using a dynamic clustering method to compress the frequent itemsets approximately in this paper. Concretely, two frequent itemsets intra-cluster similarities, *expression similarity* and *support similarity*, are defined according to the specific requirements of frequent itemsets compression. Based on the above two similarity measures, the frequent itemsets clustering criterion and its related clustering algorithm are developed. Specially, our method has two features: 1)users needn't specify the number of frequent itemsets clusters explicitly; 2)user's expectation of compression ratio is incorporated. Our initial experimental results show that our approximate frequent itemsets method is feasible and the compression quality is good.**

I. Introduction

Frequent-itemsets mining has been a hot research topic in data mining because of its applications in many important data mining tasks such as mining association rules[3], mining correlation[6], and mining sequential patterns[5] etc. The frequent-itemsets mining problem is defined as follows[3]. Given a transaction database D having N transactions, let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items, transaction t_j $(1 \leq j \leq N)$ is $t_j = \{I_{j1}, I_{j2}, \ldots, I_{jl}\}$, such that $t_j \subseteq I$. An itemset P is a set of items, such that $P \subseteq I$. The support of an itemset P is the number of transactions that contain the itemset P, denoted as $sup(P)$. The itemset P is frequent if $sup(P) \geq min\_sup$, where min_sup is a user-specified support threshold. The task of frequent itemsets mining is to find all the frequent itemsets in database D.

Frequent-itemsets mining often faces the problem of choosing the appropriate support value min_sup for mining algorithms. The frequent Itemsets generated by high support values are too obvious to be meaningful, while the frequent itemsets collection generated by low support values can be far too large to be carefully examined and understood by the users.

To reduce the output size of frequent itemsets, two major approaches have been developed for compressing purpose: lossless compression and lossy approximation. The *closed frequent itemsets* [9] and *non-derivable frequent itemsets* [7] are belong to the lossless category. The complete set of original frequent itemsets can be recovered if it is compressed by these two methods. However, the compression ratio of lossless methods is quite limited cause they emphasize too much on the supports of frequent itemsets. On the other hand, the *maximal frequent itemsets*[8], the *boundary cover sets*[2] and the *representative pattern*[10] are all lossy methods. These lossy methods have higher compression ratio but not reversible.

In addition to the drawback of information losing, the *maximal frequent itemsets* and the *boundary cover sets* just consider the expressions of itemsets while ignoring the *support* information of itemsets. The *representative pattern* method, which is proposed recently, builds up a pattern compression framework that concerns both the *expressions* and *supports* of the patterns[10]. However, two greedy algorithms of the *representative pattern* method have high computation complexity and space complexity. Concretely, the using of RP-tree (representative pattern tree) to index all the frequent patterns make them memory consuming and the coverage checking step results in the quadratic computation complexity.

So our design goal is to develop a faster algorithm, which compresses the collection of frequent itemsets approximately and has the following two features:1) both *expressions* and *supports* of frequent itemsets are used to evaluate frequent-itemsets similarity; 2) user's expectation of compression ratio is incorporated.

Our approach In this paper, we propose using a dynamic transactional clustering method to compress a collection of frequent itemsets approximately. Concretely, a modified version of transactional data clustering method, i.e. coverage density-based clustering algorithm[11] is used to partition the collection of frequent itemsets into several clusters. During the course of frequent itemsets clustering, two frequent-itemsets specific intra-cluster similarities: *expression similarity* and *support similarity* are used to evaluate the similarity of a set of frequent itemsets. Finally, the maximal frequent-itemsets of each cluster are output as compression result.

The rest of the paper is organized as follows. Section II defines two frequent-itemsets similarity measures and the frequent-itemsets clustering criterion. Section III details the frequent-itemsets clustering (compressing) procedure and analyzes the algorithm complexity. Our initial experimental results are reported in Section IV and our approximate compression approach is summarized in Section V.

978–1–4244–1674–5/08/$25.00 © 2008 IEEE

II. Problem Statement

In this section, we first analyze a concrete frequent itemsets mining example and describe the intuitive idea behind our approach. Second, the concepts of coverage density is briefly introduced and the definitions of *expression similarity* and *support similarity* of frequent-itemsets are given. Finally, the frequent-itemsets compressing (clustering) criterion is presented.

A. Analysis on A Frequent-itemsets Mining Result

Let's have a look at a concrete frequent-itemsets mining example at first. Suppose we have a transaction database D with 9 transactions shown in Table I and the user specified minimum support is $min\_sup = 2$.

After running Apriori [4] algorithm on database D, we get 13 frequent-itemsets and we use a directed graph to present the mining result, which is shown in Figure 1. In Figure 1, each graph vertex represents a frequent-itemset and includes its two parts information separated by a colon. The first part is the *expression info.* of a frequent-itemset, the second part is the *support info.* of a frequent-itemset. For example, {1,2,3}:2 means that the support of frequent-itemset {1,2,3} is 2. The directed edges among the result graph indicate the subset relationship of these frequent itemsets.

TABLE I
TRANSACTION DATABASE D

TID	Items
t1	1,2,5
t2	2,4
t3	2,3
t4	1,2,4
t5	1,3
t6	2,3
t7	1,3
t8	1,2,3,5
t9	1,2,3

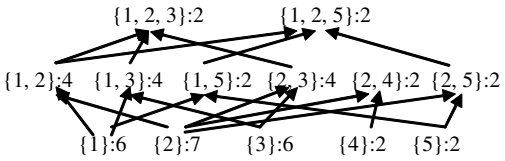

Fig. 1. Frequent-itemsets mining result of D at min_sup=2

Carefully analyzing the above frequent-itemsets mining result, we have the following observations: 1)A frequent-itemset is similar to a transaction in the view of their expression, i.e. they are also a set of items. At this point, a collection of frequent-itemsets also can be regarded as a transactional database; 2)frequent-itemsets of a mining result are highly correlated. Since Apriori algorithm searches the frequent-itemsets in a level-wise way, all the subsets of a k-frequent-itemset are also frequent [4]. For example, in Figure 1, the subsets of frequent-itemset {1,2,3} are {{1,2}, {1,3},{1},{2},{3}} and these subsets are all frequent. So in the view of expression of these frequent-itemsets, all the subsets of a frequent-itemset can be represented by their frequent superset. In the above example, frequent-itemset {1,2,3} and its subsets can be compressed into one, that is,we only report frequent-itemset {1,2,3}; 3) However, the support of a frequent-itemset and its subsets are usually different. Sometimes, the difference is very large. For example, frequent-itemset {1,2,3} has support 2 while its immediate subset {1,2} and {1,3} have support 4. If we only report {1,2,3}, then the support information of subset are lost. So in the view of *support*, frequent-itemsets {1,2,3}, {1,2} and {1,3} all should be output. Another compressive example is that frequent-itemset {1,2,5} and its subset {1,5}, {2,5}, {5} have same support. Obviously, it is sufficient to output {1,2,5} only.

Based on the above observations, we propose using a transactional clustering method to compress the collection of frequent-itemsets with specific similarity measures designed for frequent-itemsets compression application.

B. Notations

The problem of clustering frequent-itemsets can be defined as follows. Given a collection of frequent-itemsets F, let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items, F be a set of frequent-itemsets with size $|F|$, where frequent itemset f_j $(1 \leq j \leq |F|)$ is a set of items $f_j = \{I_{j1}, I_{j2}, \ldots, I_{jl}\}$, such that $f_j \subseteq I$. Let $|f_j|$ be the length of the frequent itemset f_j and $sup(f_j)$ be its support. A frequent-itemsets clustering result F^K is a partition of F, denoted by $\{F_1, F_2, \ldots, F_K\}$, where $F_1 \bigcup \ldots \bigcup F_K = F, F_i \neq \phi, F_i \bigcap F_j = \phi$.

C. Concept of Coverage Density

During the course of clustering a collection of frequent-itemsets, our method uses two similarity measures: *expression similarity* and *support similarity*. Since the *expression similarity* is based on the transactional intra-cluster similarity measure, i.e. the *Coverage Density*[11], we briefly introduce the concept of *Coverage Density* below.

To provide an intuitive illustration of our development of CD concept, let us map the transactions of D onto a 2D grid graph. Let the horizontal axis stand for items and the vertical axis stand for the transaction IDs, and each filled cell (i, j) represents the item i is in the transaction j. For example, a simple transactional dataset {abc, bc, ac, de, def} can be visualized in Figure 2.

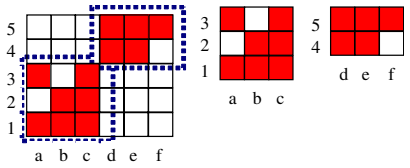

Fig. 2. An example 2D grid graph

If we look at the filled area in the graph carefully, two naturally formed clusters appear, which are {abc, bc, ac} and {de, def} indicated by two rectangles in Figure 2. In the

original graph there are 18 cells unfilled, but only 3 in the two partitioned subgraphs. The less the unfilled cells are left, the more compact the clusters are. Therefore, we consider that the problem of clustering transactional datasets can be transformed to the problem of how to obtain the minimized unfilled number of cells with appropriate number of partitions. This simple example also shows that it is intuitive to visualize the clustering structure of the transactions when they have already been ordered in the specific way as shown in the left most of Figure 2. Thus how to order and partition the transactional dataset properly is one of the key issues of clustering algorithm.

Bearing this intuition in mind, we give the definition of Coverage Density (CD).

Definition 1. Coverage Density (CD) is the percentage of filled cells to the whole rectangle area which is decided by the number of distinct items and number of transactions in a cluster.

Given a cluster C_k, it is easy and straightforward to compute its coverage density. Suppose the number of distinct items is M_k, the items set of C_k is $I_k = \{I_{k1}, I_{k2}, \ldots, I_{kM_k}\}$, the number of transactions in the cluster is N_k, and the sum occurrences of all items in cluster C_k is S_k, then the Coverage Density of cluster C_k is

$$CD(C_k) = \frac{S_k}{N_k \times M_k} = \frac{\sum_{j=1}^{M_k} occur(I_{kj})}{N_k \times M_k}. \quad (1)$$

Since the coverage density reflects the compactness of a cluster intuitively, it is used as an intra-cluster measure. Generally speaking, the larger the coverage density is, the higher the intra-cluster similarity among the transactions within a cluster.

D. Similarity Measures for Clustering Frequent-itemsets

Two similarity measures are used for clustering frequent-itemsets: *expression similarity* and *support similarity*. The *expression similarity* is a measure to evaluate the items' overlappings of a set of frequent-itemsets, while the *support similarity* is used to measure the similar degree of frequent-itemsets' supports. Actually, the *expression similarity* is the main measure in our compressing method while the *support similarity* is just a auxiliary or secondary measure. In other words, frequent-itemsets are evaluated by their *expression* at first, if they are similar, then their supports are used to further decide if they should be grouped together; if they are dissimilar at their expressions, then the *support similarity* is not used.

As described above, Coverage Density is an intuitive and ideal intra-cluster measure for transactional data[11]. So if the frequent-itemsets are treated as a group of transactions, then their expressions similarity can be calculated by a coverage-density-like formula. Below we give the formal statement of *expression similarity* computing formula:

Given a cluster (collection) of frequent-itemsets F_k having $|F_k|$ frequent-itemsets, suppose the number of distinct items is m_k, then the *Expression Similarity (ES)* of cluster F_k is

$$ES(F_k) = \frac{\sum_{j=1}^{|F_k|} |f_j|}{|F_k| \times m_k}. \quad (2)$$

Same as coverage density, the value of ES is bigger than zero but less and equal to one. For example, a one-frequent-itemset cluster's expression similarity is 1. The larger the ES is, the higher similarity among frequent-itemsets within a cluster. The ES reflects the compactness of frequent-itemsets intuitively. In addition to its intuition, the computing of our *expression similarity* is quite straightforward and fast cause it is a set-based similarity measure.

Definition 2. (*δ-frequent-itemsets-cluster*) A cluster of frequent-itemsets having $\delta\%$ expression similarity is called as δ-frequent-itemsets-cluster.

Having the concept of *δ-frequent-itemsets-cluster*, users can control the frequent-itemsets compression ratio by changing the value of δ. So in our clustering method, the δ value is an input parameter used to incorporate user's expectation.

In addition to the *expression similarity*, the *support similarity* is also used as the secondary measure in our frequent-itemsets clustering method. The purpose of *support similarity* is to preserve the support information after compression. So it would be better if each frequent-itemset within a cluster has almost the same quantity of support, i.e the distribution of these supports doesn't have much deviations from their average value. So to evaluate the *support similarity*, we directly computing the *variance* of support values of a cluster of frequent-itemsets.

Definition 3. (*support similarity*) Given a cluster (collection) of frequent-itemsets F_k having $|F_k|$ frequent-itemsets, suppose the support value of frequent-itmeset f_j is $sup(f_j)$, where $1 \leq j \leq |F_k|$ and the average support value of F_k is $E(F_k)$, then the *support similarity (SS)* of F_k is

$$SS(F_k) = \frac{\sum_{j=1}^{|F_k|} (sup(f_j) - E(F_k))^2}{|F_k|}. \quad (3)$$

The ideal *support similarity* value is zero, which means each frequent-itemset within a cluster has equal support. The larger the SS value is, the bigger deviations of support values within a cluster.

E. Clustering criterion

Having the concepts of *expression similarity* and *support similarity* for a cluster of frequent-itemsets, we define the clustering criterion based on the above two measures in this section.

Given a collection of frequent-itemsets F with size $|F|$, for a frequent-itemsets clustering result $F^K = \{F_1, F_2, \ldots, F_K\}$ where $K < |F|$ and the distinct number of items of each F_k is m_k, the clustering criterion is to maximize the *Expected Expression Similarity (EES)* value of frequent-itemsets clustering result as defined in Equation 4. At the same time, each cluster of frequent-itemsets subjects to restrict Equation 5 and Equation 6.

$$EES(F^K) = \sum_{k=1}^{K} \frac{|F_k|}{|F|} \times ES(F_k)$$
$$= \frac{1}{|F|} \times \sum_{k=1}^{K} \frac{\sum_{j=1}^{|F_k|} |f_j|}{m_k} \quad (4)$$

Subject to:

$$SS(F_k) \leq min\_ss, 1 \leq k \leq K \quad (5)$$

$$ES(F_k) \geq \delta, 0 < \delta \leq 1 \quad (6)$$

, where min_ss is a user-specified minimum *support similarity* threshold.

Our EES-based clustering algorithm tries to maximize the *EES* criterion under the restricts of user-specified min_ss value and δ value.

III. Frequent-itemsets Clustering Algorithm

Generally, a collection of frequent-itemsets is regarded as a transactional dataset and can be partitioned by a general clustering procedure like K-means' clustering procedure. However, our frequent-itemsets clustering algorithm design makes best use of two specific features of frequent-itemsets: 1)the correlations among frequent-itemsets are very high cause a frequent itemset is either a subset or a superset of other itemsets. 2)the collection of frequent-itemsets is sorted by the frequent-itemset length. According to the above two features, we designed a coverage-density-clustering-like algorithm, i.e. EES-based clustering algorithm to do frequent-itemsets compressing approximately. In this section, two problems are addressed: 1)how to partition the collection of frequent-itemsets utilizing the features of frequent-itemsets? 2)how to report the clustering result?

The EES-based clustering algorithm is a one-pass partition-based clustering algorithm, i.e. it scans the collection of frequent-itemsets one pass to assign each frequent-itemset to a cluster in terms of maximizing the EES criterion and satisfying two restricts. Concretely, it reads the frequent-itemsets from the maximal length frequent-itemsets to the shorter length frequent-itemsets sequentially. For each frequent-itemset, it is either added to an existed cluster or used to form a new cluster. The EES-based clustering algorithm tries to add the current frequent-itemset into each existed clusters to see:1) if after adding operation these clusters still satisfy the restricts; 2) adding to which cluster will maximize the EES value of current clustering result. If there is no cluster existed to satisfy the restricts, the current frequent-itemset forms a new cluster.

Obviously, for the EES-based clustering algorithm, the number of clusters is not decided by an explicit input parameter K. At this point, the EES-based clustering algorithm is a dynamic clustering method cause its number of clusters is generated dynamically through clustering procedure. Actually, the number of clusters is controlled by two user specified threshold values implicitly: δ and min_ss. The δ is direct proportional to the number of clusters while min_ss is inverse proportional to the number of clusters.

The EES-based clustering purpose is different from the general clustering purpose. Its main purpose is finding a set of representative itemsets to represent a collection of frequent itemsets. So after finishing clustering, the EES-based algorithm still need to find the representative itemsets of each cluster and report them.

The clustering result of EES-base algorithm is a group of δ-*frequent-itemsets-clusters*, each cluster may contain more than one maximal frequent itemsets. For example of a collection of frequent-itemsets in Figure 1, itemsets {1,2,3} and {1,2,5} may be partitioned in one cluster if the δ value is small. It is not reasonable if we report the combination of {1,2,3,5} as the only representative itemset of the cluster cause {1,2,3,5} is not frequent. So similar to some clustering algorithms having multiple clustering modes in each cluster, the EES-based clustering algorithm reports *multiple maximal frequent itemsets* of each cluster as output of compression result.

A sketch of the pseudo code for the EES-based algorithm is given in Algorithm 1.

Algorithm 1 EES-clustering.main()

```
Input: A collection of Frequent-itemsets F with support values; δ;min_ss
Output: Compressing result
while not end of F do
    read one frequent-itemset f from F;
    add f into existed F_i and computing the ES(F_i) and SS(F_i);
    if there is no ES(F_i) ≥ δ and SS(F_i) ≥ min_ss then
        create a new cluster F_j;
        put f into F_j;
        else
        put f into an existed cluster F_i which maximizes the current EES;
    end if
end while
for each cluster F_i do
    output the maximal frequent itemsets;
end for
```

The space consumption of EES-based clustering algorithm is quite small, since only the summary information of clusters is necessarily kept in memory. Let K stand for the number of clusters, and M stand for the maximum number of distinct items in a frequent-itemsets cluster. A total $O(K \times M)$ space is necessary for the algorithm. Even for a typical collection of frequent-itemsets with up to ten thousand distinct items, several megabytes will be sufficient for the EES-based clustering algorithm.

Since the computing of SS for each cluster is much faster compared with the ES computing for each cluster, it can be ignored. The most time-consuming part of EES-based clustering algorithm is the computing of EES values of current clustering result to find the best cluster assignment for each frquent itemset. The cost of each EES computing of current K clusters is $O(K \times |f|)$, where f is the average length of frequent itemsets. As a result, the time complexity of the whole algorithm is $O(|F| \times K \times |f|)$, where $|F|$ is the number of frequent-itemsets in dataset. Usually K and $|f|$ are much smaller than $|F|$, the running time of EES-based clustering algorithm is almost linear to the size of F. So the EES-based clustering algorithm is very fast and capable of partitioning the large collection of frequent-itemsets.

We report our initial experimental results in next section.

IV. Experimental Results

We did experiments to test:1)the feasibility of the EES-based algorithm; 2)the relationship between the compression ratio and two compression control parameters: δ and min_ss; 3) the quality of compression result.

The real datasets *mushroom* in Frequent Itemsets Mining Dataset Repository [1] is used to test the above experimental purposes. The *Mushroom* contains 8124 transactions and 119 total items. We ran Apriori [4] on *mushroom* at minimum supports $min\_sup = \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\} \times 8124$ and got 7 collections of frequent-itemsets. We modified the Apriori a little bit, which output the frequent itemsets along with their supports. We didn't use the collections at $min\_sup = \{0.1, 0.2, 1.0\} \times 8124$ because of the following reasons: 1) Apriori just outputs one frequent itemset with length 1 at $min\_sup = 8124$; 2)While we set $min\_sup = 0.2 \times 8124$, the combination explosion occurs. It's too slow to wait the outputs.

After getting 7 collections of frequent-itemsets, we ran EES-based algorithm on these collections by varying the δ from 1.0 to 0.1 and min_ss from 100 to zero. The initial results are reported below.

First, we compared the number of frequent-itemsets before and after compression. Figure 3 shows the comparison between the uncompressed number of frequent-itemsets and the compressed result at $\delta = 0.5, min\_ss = 100$. The result shows that compression effect is quite good especially at lower min_sup.

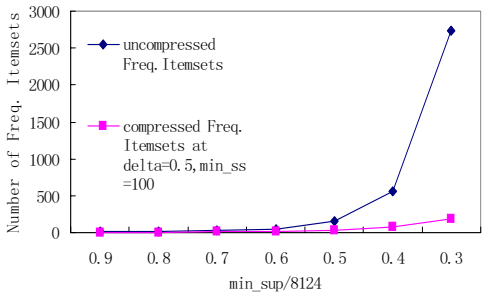

Fig. 3. Number of frequent-itemsets before and after compression

Second, we studied the relationship between the compression ratio and two control parameters. We selected the collection of Frequent Itemsets at $min\_sup = 0.8 \times 8124$ as testing dataset, which has 23 frequent itemsets. The results are shown in Figure 4 and 5. The two graph results prove our analysis before, i.e. the δ is direct proportional to the number of compressed frequent itemsets while the min_ss is inverse proportional to the number of compressed frequent itemsets. In other words, the δ is inverse proportional to the compression ratio and the min_ss is direct proportional to the compression ratio.

Third, we analyzed the compression results to evaluate the quality of EES-based clustering algorithm. Although the EES-based algorithm just scans the collection of frequent-itemsets one pass, the quality of clustering result is quite good. Below, we report the clustering result of the $min\_sup = 0.8 \times 8124$ collection of frequent-itemsets at $\delta = 0.5, min\_ss = 100$ in Table II. According to Table II, the final compression result is: $\{34, 36, 85, 86\}$, $\{34, 85, 86, 90\}$, $\{34, 85, 86\}$, $\{36, 85\}$, $\{85, 90\}$ and $\{85\}$. Obviously, the quality of such compression result is good.

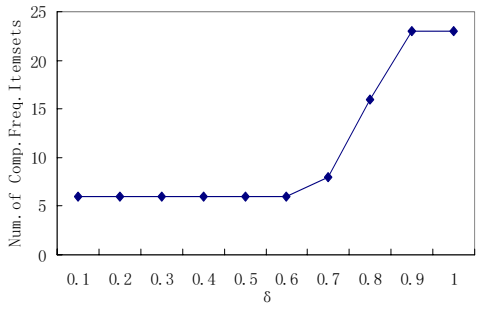

Fig. 4. δ vs. the compressed number of frequent-itemsets

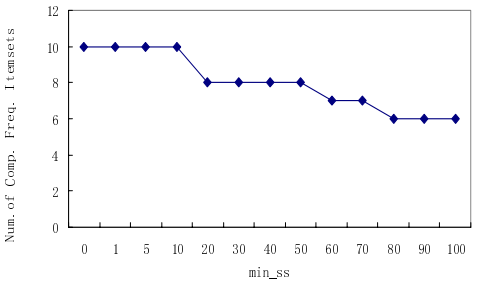

Fig. 5. min_ss vs. the compressed number of frequent-itemsets

TABLE II
Clustering result of $min\_sup = 0.8$ collection of frequent-itemsets

Cluster 1	Cluster 2	Cluster 3
{34,36,85,86}:6602	{34,85,86,90}:7288	{34,85,86}:7906
{34,36,85}:6602	{34,85,90}:7296	{34,85}:7914
{34,36,86}:6602	{34,86,90}:7288	{34,86}:7906
{36,85,86}:6620	{85,86,90}:7288	{85,86}:7924
{34,36}:6602	{34,90}:7296	{34}:7914
{36,86}:6620	{86,90}:7288	{86}:7914
Cluster 4	Cluster 5	Cluster 6
{36,85}:6812	{85,90}:7488	{85}:8124
{36}:6812	{90}:7488	

The above initial experimental results show that the EES-based algorithm is capable of compressing the collection of frequent itemsets. Although it is an approximate compression method but compression quality is high.

V. Conclusion

In this paper, we propose using a dynamic clustering method to compress the frequent itemsets approximately, i.e the collection of frequent itemsets is partitioned into clusters at first, then the multiple *maximal frequent itemsets* of each cluster are

output as the compression result. Two frequent itemsets intra-cluster similarities: *expression similarity* and *support similarity* are defined in this paper. Based on the two similarity measures, the frequent itemsets clustering criterion and its related frequent itemsets clustering algorithm are also developed. Our initial experimental results show that our clustering method obtains not only high compression ratio but also high-quality compression result.

ACKNOWLEDGMENT

This work was supported by Chinese 863 High-Tech Program under Grant 2007AA01Z321.

REFERENCES

[1] Frequent itemset mining dataset repository. *http://fimi.cs.helsinki.fi/data/*.

[2] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. *Proceedings of Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 12–19, 2004.

[3] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. pages 207–216, 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 12–15 1994.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. pages 3–14, 1995.

[6] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. *Proc. of ACM SIGMOD Conference*, pages 265–276, 1997.

[7] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, 2431:74–85, 2002.

[8] D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning (extended abstract). *Proc. of ACM PODS Conference*, pages 209–216, 1997.

[9] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *The 7th International Conference on Database Theory*, 1540:398–416, 1999.

[10] D. Xin, J. Han, X. Yan, and H. Cheng. On compressing frequent patterns. *Data Knowl. Eng.*, 60(1), 2007.

[11] H. Yan, K. Chen, L. Liu, J. Bae, and Z. Yi. Efficiently clustering transactional data with weighted coverage density. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)*, pages 367–376, November 2006.