1-2009

Determining the Best K for Clustering Transactional Datasets: A Coverage Density-based Approach

Hua Yan
Wright State University - Main Campus

Keke Chen
Wright State University - Main Campus, keke.chen@wright.edu

Ling Liu

Determining the Best K for Clustering Transactional Datasets: A Coverage Density-based Approach

Abstract

The problem of determining the optimal number of clusters is important but mysterious in cluster analysis. In this paper, we propose a novel method to find a set of candidate optimal number Ks of clusters in transactional datasets. Concretely, we propose Transactional-cluster-modes Dissimilarity based on the concept of coverage density as an intuitive transactional inter-cluster dissimilarity measure. Based on the above measure, an agglomerative hierachical clustering algorithm is developed and the Merge Dissimilarity Indexes, which are generated in hierachical cluster merging processes, are used to find the candidate optimal number Ks of clusters of transactional data. Our experimental results on both synthetic and real data show that the new method often effectively estimates the number of clusters of transactional data.

Key words: Transactional-cluster-mode, Transactional-cluster-modes Dissimilarity, Merging Dissimilarity Index, Differential MDI curve

1 Introduction

Clustering is an important tool in data analysis. It uses data similarity measures to partition a large dataset into a set of disjoint data clusters such that data points within the clusters are close to each other and the data points from different clusters are dissimilar from each other in terms of the similarity measure used. There have been a lot of data clustering algorithms developed in recent years including numeric data clustering and categorical data clustering [1,3–6,8,9,11–14,16,18,19,**?**,20]. However, the problem of determining the number of clusters is still pending. Most of the algorithms mentioned above need a user-specified number of K clusters or implicity cluster number control parameters in advance. For numerical data clustering, the inherent distance

feature is utilized and the number of clusters can be validated through geometry shape or density distribution [5,**?**,12,17]. The techniques used in numerical data clustering are not suitable for categorical data clustering because of the lack of the distance meaning for the categorical data. [9] develops a visualization tool for clustered categorical data to find the optimal number of clusters by involving human subjective judgement. But it cannot find the optimal number of clusters automatically.

Recently the Best K method BKPlot has been developed at Georgia Tech [**?**]. The BKPlot method studies the entropy difference between the clustering structures with varying K and reports only those Ks where the clustering structure changes dramatically as the candidate best Ks, which greatly reduces the search space of finding the domain-specific candidate best Ks. A hierarchical entropy-based algorithm ACE proposed in [**?**] to generate high-quality approximate BKPlot, which can capture the candidate best Ks with small errors.

Transactional data is a kind of special categorical data, which can be transformed to the traditional row by column table with Boolean values. Usually the transformed dataset has two features: large volume and high dimensionality. For instance, a market basket dataset may contain millions of transactions and thousands of distinct items, although each transaction usually contains only about tens of items. The transformation to Boolean data increases the dimensionality from tens to thousands. Can the BKPlot find the significant clustering structures in transactional data? The SCALE framework for clustering transactional data in [**?**] integrates the BKPlot to assess the number of clusters in transactional datasets and help clustering free from parameter tuning. The experimental results in [**?**] show the predefined class numbers are usually included in the candidate cluster numbers provided by the BKPlot. However, we noticed two weaknesses of BKPlot on dealing with transactional datasets.

First, the BKPlot produces noisy candidate cluster numbers even in a very well-structured transactional dataset. For example, the BKPlot recommends {2,5} for transactional dataset {1100000000, 0011000000, 0000110000, 0000001100, 0000000011}, where "1100000000" means the first two items are in the first transaction. Similarly, for a one layer structure transactional dataset, which has 1000 items and 20 clusters defined with a structure similar to Figure 11, the candidate cluster numbers are {2, 5, 10, 20}. Obviously, these {2,5,10} are not significant structures.

Second, the BKPlot becomes time-consuming while BKPlot deals with transaction datasets with large number of items. According to [**?**], the time complexity of hierarchical entropy-based ACE, which is used to generate approximate BKPlot, is $O(dmN^2logN)$. In the above time complexity expression, d is the

dimension of dataset, m is the average column cardinality, and N is the number of records. Analyzing the above time complexity expression, we can see parameter d also has great influence on the efficiency of ACE algorithm especially when d is extremely large in the case of transactional data. BKPlot will be very time-consuming when it is applied to the transactional data, which has to be transformed to high dimensional Boolean data in advance.

Our goal is to design a new method especially for transactional data, which can find significant clustering structures with less noisy candidates and more efficiently. We first propose the concept of Transactional-cluster-modes Dissimilarity based on the concept of Coverage Density [**?**], which intuitively reflects the inter-cluster dissimilarity of transactional data. Then we use this dissimilarity measure to develop an agglomerative hierachical transactional clustering algorithm ACTD (Agglomerative Clustering algorithm with Transactional-cluster-modes Dissimilarity). Since the Transactional-cluster-mode of a cluster is just a subset of cluster items, only part of items are involved in the computation of dissimilarity of two modes. Suppose the average length of a cluster mode is M, then the cost of Transactional-cluster-modes Dissimilarity computation is $O(M)$, which is often hundreds of times faster than the entropy computation cost factor $O(dm)$ used in BKPlot method. In the course of ACTD clustering, a set of MDI (Merging Dissimilarity Indexes) values is generated during the hierachical cluster merging process. Similar to the rationale behind BKPlot, where the cluster dissimilarity changes dramatically, the significant clustering structure emerges. We analyze these MDI values and plot the differential MDI (DMDI) curve to identify the candidates of the optimal number of clusters. Our experimental results on both synthetic data and real data show that the DMDI curve effectively indicates the significant clustering structures of transactional data with higher quality result.

The rest of the paper is organized as follows. Section 2 gives the notations used in this paper and defines an intuitive concept of inter-cluster dissimilarity, i.e. Transactional-cluster-modes Dissimilarity. The SCALE framework and the concept of Coverage Density are also briefly introduced in this section. Section 3 details the agglomerative hierarchical clustering algorithm and the differential MDI curves used for identifying the significant clustering structures. We report our initial experimental results in Section 4 and summarize our approach in Section 5.

2 Transactional-cluster-modes Dissimilarity

2.1 Notations

We first define the notations of transactional dataset and transactional clustering result used in this paper. A transactional dataset D of size N is defined as follows. Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items, D be a set of N transactions, where transaction t_j $(1 \leq j \leq N)$ is a set of items $t_j = \{I_{j1}, I_{j2}, \ldots, I_{jl}\}$, such that $t_j \subseteq I$. Let $|t_j|$ be the length of the transaction. A transaction clustering result C^K is a partition of D, denoted by $\{C_1, C_2, \ldots, C_K\}$, where $C_1 \bigcup \ldots \bigcup C_K = D, C_i \neq \phi, C_i \bigcap C_j = \phi$.

2.2 Overview of the SCALE Framework

We briefly describe the design of SCALE (Sampling, Clustering structure Assessment, cLustering and domain-specific Evaluation)[**?**], a fully automated transactional clustering framework, before we discuss in detail the design and implementation of our new clustering structure assessment method. The SCALE framework is designed to perform the transactional data clustering in four steps as shown in Figure 1.

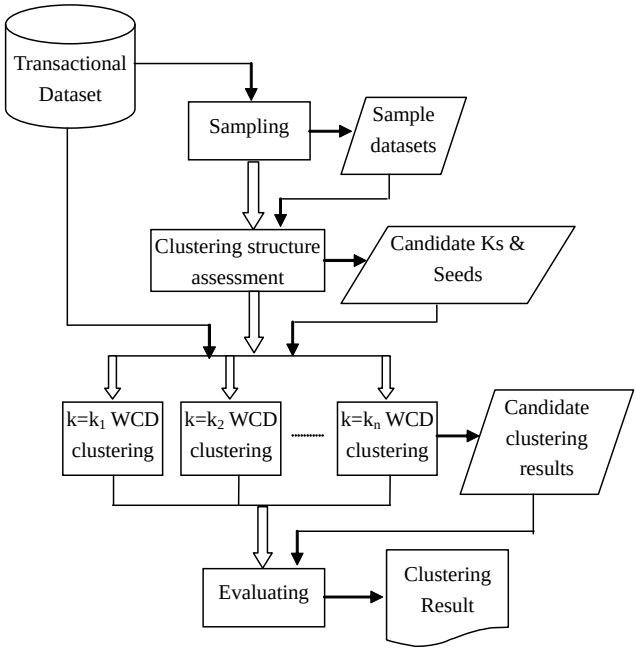

Fig. 1. The SCALE framework

SCALE uses the sampling step to handle large transactional dataset. Standard

sampling techniques are used in the sampling step to generate some sample datasets from the entire large dataset. The framework assumes the primary clustering structure (with small number of large clusters) is approximately preserved with appropriate sample size.

In the clustering structure assessment step, SCALE determines the candidates of significant clustering structure and generates the candidate "best Ks" based on sample datasets. In our conference version prototype implementation, the BKPlot method [**?**] is integrated directly into the SCALE and the hierarchical algorithm ACE [**?**] is used to generate high-quality approximate BKPlots graph, which can capture the candidate best Ks with small errors. The experimental results in [**?**] show that the predefined class numbers are usually included in the candidate cluster numbers provided by the BKPlot.

However, the ACE becomes time-consuming on dealing with transactional datasets with large number of items at this step. In this paper, we design a transactional data specific method, i.e. the DMDI method, to do clustering structure assessment. Similarly, a hierarchical algorithm ACTD, which is based on the concept of Transactional-cluster-modes Dissimilarity, is developed to generate DMDI curves. The experimental results show that ACTD is faster than ACE in dealing with transactional datasets with large number of items.

Both ACE and ACTD also generate a hierarchical clustering tree, where the cluster seeds can be found at different Ks. The clustering structure assessment step outputs the best Ks and the cluster seeds at the best Ks to the clustering step.

The clustering step applies the WCD [**?**] clustering algorithm to perform initial cluster assignment. The initial assessment result is then used to guide the WCD clustering over the entire dataset in an iterative manner until no transaction is moved from one cluster to another in one pass with respect to maximizing WCD. At the end of iterative assignment refinement, a small number of candidate clustering results are generated. Finally, we use the domain-specific measures (AMI and LISR)[**?**] to evaluate the clustering quality of the candidate results produced in the clustering step and select the best one.

We call the SCALE using BKPLot method as *SCALE-ACE* and the SCALE using DMDI method as *SCALE-ACTD*. In this paper, we mainly present our design principles of DMDI method and algorithmic details of ACTD algorithm. And our experimental results are the performance and quality comparison of two frameworks in their clustering structure assessment step.

In this section we briefly introduce the concept of Coverage Density (CD)[**?**]. 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.

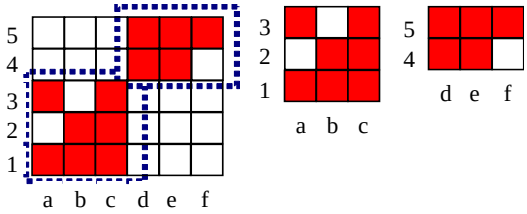

Fig. 2. An example 2D grid graph

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.

2.4 Concept of Transactional-cluster-modes Dissimilarity

Besides the intra-cluster similarity measure, the inter-cluster dissimilarity is also used to measure the quality of clustering results. In this paper, we propose the concept of Transactional-cluster-modes Dissimilarity as the inter-cluster dissimilarity measure in transactional data clustering. Transactional-cluster-modes Dissimilarity is based on the concept of Coverage Density. Our experiments show that the Transactional-cluster-modes Dissimilarity is an efficient measure in doing hierachical clustering and finding the optimal candidate k number of clusters. In the following, we define the concept of Transactional Cluster Mode at first.

Definition 2. Transactional Cluster Mode is a subset of cluster items, where the occurrence of each item in the cluster is above the user-specified proportion of transactions. Given a transactional cluster C_k having N_k transactions and M_k distinct items, suppose the user-specified minimum support is θ, then the transactional cluster mode CM_k of cluster C_k is $CM_k = \{I_{kj} | occurr(I_{kj}) \geq (N_k \times \theta), 1 \leq j \leq M_k\}$. The length of cluster mode CM_k is $|CM_k|$.

Definition 3. Transactional-cluster-modes Dissimilarity is the dissimilarity between two transactional cluster modes. Given a pair of clusters C_i and C_j, suppose the cluster-modes of two clusters are CM_i and CM_j respectively, then the Transactional-cluster-modes Dissimilarity between the C_i and C_j is

$$d_m(C_i, C_j) = 1 - CD(CM_i \bigcup CM_j) \tag{2}$$

Simplifying the above formula, we get $d_m(C_i, C_j) = 1 - \frac{|CM_i|+|CM_j|}{2\times|CM_{ij}|}$, where $|CM_{ij}|$ is the number of distinct items after merging two cluster modes and thus $|CM_{ij}| \geq \max\{|CM_i|, |CM_j|\}$. Here, it is easy to conclude that $d_m(C_i, C_j)$ is a real number between 0 and $\frac{1}{2}$. Not surprisingly, when two cluster modes have the same set of items, that is $CM_i = CM_j = CM_{ij}$, $d_m(C_i, C_j) = 0$. When two complete different cluster modes having no overlapping between the sets of items, that is $CM_{ij} = CM_i + CM_j$, merging them will result in a maximum dissimilarity $d_m(C_i, C_j) = \frac{1}{2}$. When two cluster modes with overlapping between the sets of items, the dissimilarity will be a real number between 0 and $\frac{1}{2}$ and the concrete value is decided by the level of overlapping. Three examples are given to illustrate the above situations in Figure 3, Figure 4 and Figure 5.

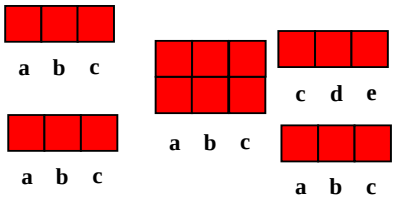

Fig. 3. Two cluster modes with zero dissimilarity

Fig. 4. Two clusters modes with positive dissimilarity

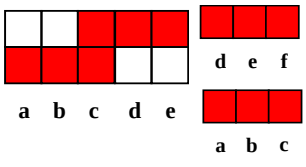

Fig. 5. Two clusters modes with max dissimilarity

The Transactional-cluster-modes Dissimilarity reflects the inter-cluster dissimilarity intuitively, i.e., when the two clusters are similar in structure, merging them will not bring large structural change into the partition, thus, the dissimilarity between cluster modes will be small; when the two clusters are very different, merging them will bring large structural change into the partition, thus, the dissimilarity between clusters will be large. Therefore, we say the above measure evaluates the structural difference between clusters and the Transactional-cluster-modes Dissimilarity is an ideal inter-cluster dissimilarity measure for transactional data.

3 Finding Significant Clustering Structure in Transactional Datasets

A lot of clustering algorithms have been developed for different application areas. These algorithms usually are designed under the assumption that datasets have some clustering tendency. However, the clustering process is unsupervised, without predefined classes or template examples. Consequently, the clustering results of these clustering algorithms are greatly decided by the features of datasets and the input parameter values [17].

The optimal clustering structure is the partition that best fits the inherent structure of the dataset. Finding the optimal clustering structure is one of the most important but difficult problem. First, the guidance for choosing the appropriate input parameter values is often skipped by most clustering literatures. Obviously, if the number of clusters is assigned to an improper value, it is impossible for the clustering algorithms to get the optimal result. Second, datasets may also have multiple optimal numbers of clusters, according to different levels of clustering granularity, which may be preferred by different application cases. It is also challenging to find all of them.

Before proceeding to our algorithm, we first define the concept of *Optimal Clustering Structure* and *Significant Clustering Structure*.

Definition 4. An optimal clustering structure for K number of clusters is the one that optimizes certain clustering criteria. A significant clustering struc-

ture is the most important clustering structure among all optimal clustering structures with different K.

One significant clustering structure corresponds to one optimal number of clusters. Note that there might be multiple significant clustering structures according to different levels of clustering granularity. It would be extremely difficult to find all optimal clustering structures for different K in order to determine the significant clustering structures, due to the NP-hard complexity of the clustering problem. However, we are able to identify the best number of clusters with approximation algorithms. In this section, we provide the design details of the ACTD algorithm that is used to efficiently find best K for transactional datasets.

3.1 ACTD: Agglomerative Clustering algorithm with Transactional-cluster-modes Dissimilarity

Having Transactional-cluster-modes Dissimilarity as the measure of inter-cluster dissimilarity, we develop an agglomerative clustering algorithm ACTD to do transactional data clustering and find the significant clustering structures of dataset. With the common agglomerative clustering process, we briefly describe the ACTD algorithm.

ACTD uses bottom-up process to do clustering. It begins with the scenario where each transaction is a cluster. Then, an iterative process is followed: in each step, the algorithm finds a pair of clusters C_i and C_j that are the most similar, i.e., the $d_m(C_i, C_j)$ of C_i and C_j is the minimum value among all dissimilarity values of possible pair of clusters. The algorithm merges the pair clusters C_i and C_j and records the $d_m(C_i, C_j)$ as Merging Dissimilarity Index (MDI) of current clustering structure. The iterative process is stopped until there are just two clusters left. Finally the algorithm outputs all clustering results and MDIs generated during the hierachical cluster merging process.

In order to efficiently implement the ACTD, we maintain three tables: clusters summary table $T_S[\ ]$ is used for recording the basic information of each cluster of current clustering result. The basic information of cluster C_k include the number of transactions $T_S[k].N_k$, the sum occurrences of items $T_S[k].S_k$, the number of distinct items $T_S[k].M_k$, and the occurrences of each item $T_S[k].occur(I_{kj})$. At the beginning, the size of summary table is the number of transactions of the input dataset. In the iterative merging procedure, the space used by the merged clusters is freed; The second table is pair-clusters dissimilarity table $T_d[\ ][\ ]$, which is used for computing the modes dissimilarity of each pair clusters of current clustering result. For example, $T_d[i][j]$ keeps the value of $d_m(C_i, C_j)$. It is a symmetric table. The algorithm searches the

$T_d[\ ][\ ]$, finds and merges the the pair clusters with minimum dissimilarity. The third table is Merging Dissimilarity Indexes Table $T_{MDI}[\ ]$, which keeps track of the minimum transactional-cluster-modes dissimilarity of all merging operations.

Algorithm 1 shows the sketch of the main procedure of ACTD. For the merging pair clusters, the algorithm chooses one cluster with smaller cluster number as the master cluster and the other as merged cluster. The merging operation need copy the merged cluster info into the master cluster and invalidate the merged cluster. The detailed merging operation algorithm is described in Algorithm 2. After each merging operation, there is an updating operation. The updating operation mainly recalculates the pair-cluster modes dissimilarity between newly formed cluster and the other valid clusters. Its algorithm description is given in Algorithm 3.

Algorithm 1 ACTD.main()

Input: Transactional dataset D, Number of transactions N, Min-Support θ
Output: $T_{MDI}[]$ and all clustering results
Initialize $T_S[N]$, $T_d[N][N]$,$T_{MDI}[N]$;
$clusterNo = N$;
while $(clusterNo > 1)$ **do**
  $T_{MDI}[clusterNo]$ = min { $T_d[i][j]$, $i \neq j$};
  merging (i, j);//Algorithm 2
  updating (i, j);//Algorithm 3
  $clusterNo - -$;
end while
output $T_{MDI}[]$ and all clustering results;

Algorithm 2 ACTD.merging(i,j)

```
void merging (i, j)
{
T_S[i].N_i+ = T_S[j].N_j;
T_S[i].S_i+ = T_S[j].S_j;
for (i = 0; i < T_S[j].M_j; i + +) do
  if (T_S[j].I_ji not exist in T_S[i].I) then
    T_S[i].M_i + +;
    T_S[i].occur(I_ik) = T_S[j].occur(I_ji);
  else
    T_S[i].occur(I_ik)+ = T_S[j].occur(I_ji);
  end if
end for
}
```

The cost of transactional-cluster-modes dissimilarity computing is $O(M)$ if the average length of transactional cluster mode is M. The most time-consuming part of our algorithm is updating the $T_d[\ ][\ ]$, and the time complexity of

Algorithm 3 ACTD.updating(i,j)

```
void updating (i, j)
{
free T_S[j];
Invalidate T_d[j].[*];
for (each valid cluster u) do
   Recalculating T_d[i][u];
end for
}
```

this part is $O(N^2 logN)$ in the worst case. So the overall time complexity is $O(MN^2 logN)$. Compared to the factor dm in the complexity of BKPlot algorithm, $O(dmN^2 logN)$, M is often hundreds times smaller. For a common transactional dataset, dm will be around thousands to tens of thousands, while the length of transaction clustering mode M is often around tens. Therefore, the improvement in terms of time complexity is very significant. The space complexity is around the same level of BKPlot algorithm. The clusters summary table $T_S[\,]$ needs $O(MN)$ space, the Merging Dissimilarity Indexes Table $T_{MDI}[\,]$ requires $O(N)$ space, and pair-clusters dissimilarity table $T_d[\,][\,]$ costs $O(N^2)$ space respectively.

3.2 Plotting Differential MDI Curve to Find Significant Clustering Structure

Traditionally, statistical validity indexes based on geometry and density distribution are applied in clustering numerical data [17]. A typical index curve consists of the statistical index values for different K number of clusters. The Ks at the peaks, valleys, or distinguished "knees" on the index curve, are regarded as the candidates of the optimal number of clusters. Are there such index curves indicating the significant clustering structures for transactional data as well?

We plot all the MDI values in Table T_{MDI}, which is outputted by algorithm ACTD and our results show the curves of MDI values are usually a decreasing curve with some plateaus (Figure 7), i.e., some different Ks have same MDI value. Intuitively, these plateaus indicate similar structural changes caused by the consecutive merging operations. But from plateau to plateau, the MDI values change greatly, which indicates more significant structural changes happening compared to the neighboring changes. A conceptual demonstration of "agglomerative clustering procedure" in Figure 6 can help to understand the reason of MDI curve shape.

In Figure 6, the first three merging operations are done on the very similar clusters, so the MDI values are very small and roughly equal. Then there will be a plateau in the MDI curve. But the fourth merging operation will change

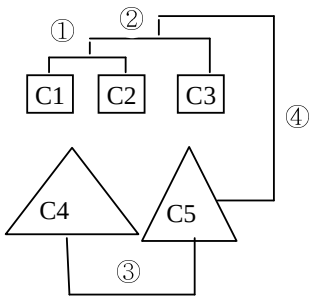

Fig. 6. $MDI[1] \approx MDI[2] \approx MDI[3]$, but $MDI[4] \gg MDI[3]$

the clustering structure dramatically, so the MDI value of the fourth merging operation is much larger than the previous one. It might become the start point of another plateau if another clustering structure exists. Here we would like to give the definition of Differential MDI below.

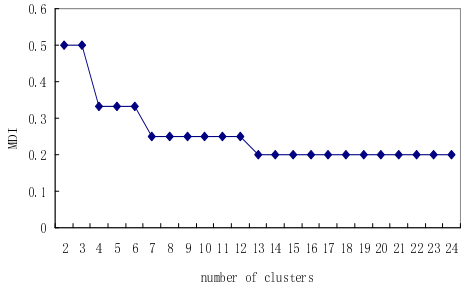

Fig. 7. MDI curve of lenses at Min–support=0.8

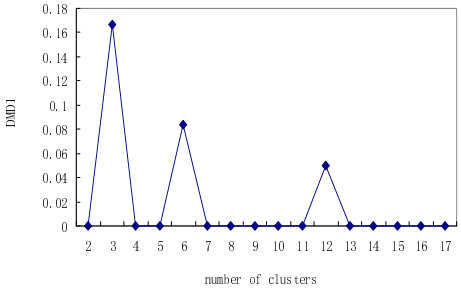

Fig. 8. Differential MDI curve of lenses at Min-support=0.8

Definition 5. For any two neighboring clustering results C^k and C^{k+1}, the differential MDI is $DMDI(k) = MDI(k) - MDI(k-1)$.

A small $DMDI$ means the merging operations of two neighboring partition schemes were merging similar clusters and didn't change the structure dramatically. A big $DMDI$ means the merging operation might cause the change of the clustering structure. We plot these Differential MDI values and get a Differential MDIs Curve. The critical points among the plateaus become the peaks of Differential MDIs Curve as shown in Figure 8. So the Differential MDIs Curve is the index curve we are looking for to find the candidate optimal number of clusters. Our experimental results in Section 4 will show the feasibility of Differential MDIs Curve for finding significant structures of transactional datasets.

A DMDI curve may have more than one peak as shown in Figure 8. Then which peak indicates the right number of clusters? We think that the right number

of clusters of a dataset is not unique especially when the clustering structure is hierarchical. For example, the dataset shown in Figure 12 is a well-structured two-layer dataset. The top layer has 5 clusters and four of which have two overlapped sub-clusters. So both k=5 and k=9 are right number of clusters of the dataset in different clustering granularity. In addition, some candidate optimal cluster numbers indicated by some peaks may not consistent with the predefined classes of datasets. We call these Ks as additional Ks. Although these additional Ks are not consistent with the domain knowledge, they may be the valuable Ks to be utilized to explore more hidden knowledge.

3.3 Assessing the Cluster Number of Large Transactional Datasets

For the problem of clustering large transactional datasets, we suggest using SCALE-ACTD framework. Under the SCALE-ACTD framework, we do sampling on original large datasets at first, then run ACTD on a group of sample datasets to find the significant clustering structures.

Running ACTD on a group of sample datasets will bring one problem, that is, are these DMDI curves of sample datasets identical with the DMDI curve for the original dataset? Here, we would like to call the DMDI curves of sample datasets as **Sample DMDI curves** and the DMDI curve of original large dataset as **Original DMDI curve**. Usually, sample DMDI curves converge to the original DMDI with the increase of sample size. Our experimental results on sample datasets show that sample DMDI curves will deviate from the original DMDI curve if the sample size is small. How many samples are needed to guarantee the consistency between the clustering structure in the sample dataset and that in the original dataset? Some literatures [15,6] gave rough estimation, which related to the cluster density and the cluster distribution.

In practice, we will try as many sample points as possible that the assessment algorithm can handle in amount of acceptable time. Note that it is always beneficial to include large sample set, in order to minimize the potential errors in assessment. Therefore, any improvement to the performance of assessment algorithm, as the performance boosting of ACTD to ACE that is used in the BKPlot method, will be preferred.

4 Experiments

We did experiments on both synthetic datasets and real datasets to test if: 1) the SCALE-ACTD is faster than SCALE-ACE in clustering structure assessment step if the dataset has a large number of items; 2) the DMDI method

can identify all best Ks for experimental datasets; 3) the DMDI method provides less noisy candidate cluster numbers than the BKPlot method on well-structured datasets;4) the DMDI method is robust enough to handle the datasets with noise transactions; 5) the clustering quality of our SCALE is better than CLUTO (a document clustering software package) on dealing with transactions.

Before we show our experimental results, we introduce the symbols used to annotate the synthetic datasets first. The three primary symbols are the average transaction length T, the total number of items I and the number of transactions D. For a dataset having $T = 10$, $I = 1000$ and $100K$ transactions is denoted as $T10I1000D100K$. When we use the data generator in [2] to generate synthetic transactional data, the average size of patterns is 4 and the number of patterns is 2000 if there is no specific description.

4.1 Performance Evaluation

We did performance evaluation experiments to verify our analysis on time complexity of two algorithms ACE and ACTD. First, we studied the two performance critical factors for ACTD: the number of transactions N and the average length of transactions T, which is highly correlated to the average length of transactional mode M. Then we compared the performance of the two algorithms on a set of datasets. The detail information of datasets are given below:

TxI1000D1k series Data generator in [2] is used to generate synthetic transactional datasets for studying the ACTD time-complexity factor M. Since the purpose of our experiments is studying the relationship between the running time and the average transaction length T, we generated seven datasets from $T5I1000D1K$ to $T400I1000D1K$ by setting the average transaction length from 5 to 400.

Retail A real large market basket dataset [7] contains 88162 transactions and 16470 items, which is approximately 5 months receipts being collected. The average number of distinct items purchased per receipt is 13 and most customers buy between 7 and 11 items per shopping visit. Retail's sample datasets with different sizes were used to study the ACTD time-complexity factor N and to compare the performance of two algorithms.

We ran both ACTD and ACE on TxI1000D1k series datasets, the running time for different sizes are shown in Figure 9. The ACTD curve in Figure 9 shows that for small average transaction length $T < 100$, the running time of ACTD is almost a constant, while for $T > 100$, the running time of ACTD becomes linear to the average transaction length T. This shows that the aver-

age length of transactional cluster mode can be quite stable even though the transaction length varies in the range (0, 100]. Since the average transaction length for most real transactional datasets is less than 100, the effect of average transaction length keeps constant on the time efficiency of ACTD. On the other hand, although the running time of ACE is not sensitive to the factor of average length of transaction, the ACE curve in Figure 9 shows that the total time cost of ACE is almost 10 times higher than that of ACTD.

We sampled Retail with different sizes from 100 to 2000 and these sample datasets were transformed into categorical datasets with 16470 dimensions before running ACE. The time spent on these sample datasets are shown in Figure 10. We can see that ACTD is much faster than ACE. The performance test results confirm our analysis on the time complexity of ACE and ACTD: the large number of dimensions d has great impact on running time of ACE algorithm since all transactions have to be transformed to Boolean data, while only the average transaction length has influence on ACTD algorithm. Since the average transaction length is usually much smaller than the number of total items in a real transactional dataset, ACTD is much faster than ACE on dealing with a common transactional dataset.

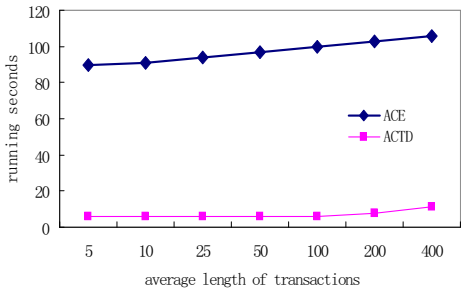

Fig. 9. The running time of ACTD on TxI1000D1K series with varying average transaction lengths

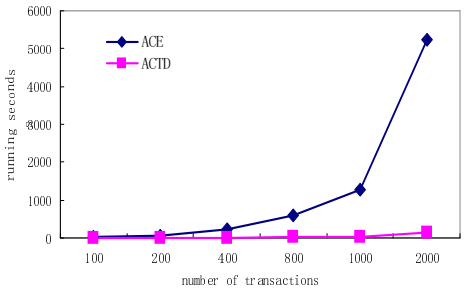

Fig. 10. The running time of ACE and ACTD on Retail with varying number of transactions

4.2 Quality Evaluation

Our experiments have used seven small datasets to evaluate the quality of the DMDI method and the BKPlot method on transactional datasets. Two small synthetic transactional datasets are constructed by ourselves: T50I1000D200 and T6I46D200, which have clearly verifiable clustering structure and manually predefined the class label of each transaction. In addition, we used five real (non-transactional) datasets: Lenses and Soybean-small are from the UCI machine learning repository [1]; Tr41, Wap and LA1 are from the documents

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

clustering datasets [2] .

The detail information of these seven datasets are given below.

T50I1000D200 is generated with one-layer clustering structure as shown in Figure 11. T50I1000D200 has 200 transactions and each transaction has 50 items. It has 20 clusters in the same size, i.e., each cluster has 10 transactions and transactions in the same cluster have completely same items.

T6I46D200 has two-layer clustering structure as shown in Figure 12. T6I46D200 has 200 transactions and the length of each transaction is 6. The total number of items is 46. The top layer has 5 clusters with 40 transactions in each cluster, four of which have two overlapping sub-clusters of 20 transactions. In both Figure 11 and Figure 12, the non-blank areas represent transactions having these items.

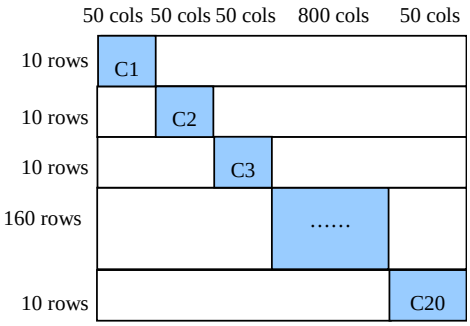

Fig. 11. Structure of one layer synthetic data

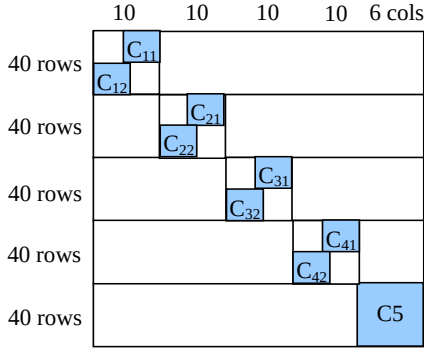

Fig. 12. Structure of two layer synthetic data

Lenses Lenses is a small categorical dataset for fitting contact lenses with 24 instances. Each instance has 4 attributes describing the eye condition of patients and a predefined class label. It has three types of class labels indicating the type of lenses the patient should wear.

Soybean-small is a categorical dataset used to classify the soybean diseases. The dataset has 47 records and each record has 35 attributes describing the features of the plant. There are four predefined classes in the dataset.

Tr41 were derived from TREC-5, TREC-6, and TREC-7 collections [3] . It has 878 documents and 7454 terms. The predefined class number is 10.

Wap are from the WebACE project and contains 1560 documents. Each document corresponds to a web paper listed in the subject hierarchy of Yahoo!.

[2] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download
[3] http://trec.nist.gov

The total number of terms is 8460 and the predefined class number is 20.

LA1 was obtained from the articles of the Los Angeles Times that was used in TREC-5. It includes the documents from the entertainment, financial, foreign, metro, national and sports, i.e., the predefined class number is 6. The number of documents is 3204 and the total number of terms is 31472.

Before running ACTD algorithm, these two categorical datasets, i.e. Lenses and Soybean-small are transformed to transaction datasets. The procedure of transformation is: 1) assign an item number to each categorical attribute value; 2) read the categorical dataset row by row and covert each attribute value into its corresponding item number; 3) write back the transformed item number into a new dataset file row by row. As a result, Lenses is transformed into a transactional dataset with 9 items and Soybean-small becomes a 72 items transactional dataset.

Since the last three datasets originally are document datasets, whose term frequency information is also included in the datasets. Before running the ACTD, these three datasets are also transformed into transactional datasets by removing the term frequency information.

Similarly, before running ACE algorithm, five transaction datasets, i.e., two synthetic transaction datasets and three transformed documents datasets, are transformed to Boolean dataset. Suppose there is N transactions and total $|I|$ items in a transactional dataset, the transformation procedure is: 1) construct a $N \times |I|$ matrix; 2) the value of matrix element (i, j) is 1 if the ith transaction has item j, otherwise the value of matrix element is 0. The row and column numbers of these transformed boolean datasets are summarized in Table 1.

Table 1
Summary for five transformed boolean datasets

Datasets	#class	#row	#column
T50I1000D200	{20}	200	1000
T6I46D200	{5, 9}	200	46
Tr41	{10}	878	7454
Wap	{20}	1560	8460
LA1	{6}	3204	31472

Three measures are used to evaluate the quality of DMDI curves and BKPlot curves. First, the predefined number of classes is used as the expected value for evaluating the quality of two index curves on identifying clustering structures. Second, two measures proposed in [**?**], i.e. Coverage Rate (CR) and False Discovery Rate (FDR), are used to compare the quality of DMDI curves and BKPlot curves. Below we give brief definitions of two measures.

Coverage Rate The Coverage Rate (CR) is the percentage of inherent significant Ks indicated by detecting methods. There could be more than one

significant clustering structures for a particular dataset. A robust detecting method should always include all of the significant Ks.

False Discovery Rate The False Discovery Rate(FDR) is the percentage of the noisy candidate Ks indicated by detecting methods. There could be some Ks, which are actually not critical but suggested by detecting methods. In order to efficiently find the most significant ones, we prefer a detecting method to have less false candidate Ks as possible.

The results of two synthetic datasets and five real datasets are summarized in Table 2. Table 2 shows: 1) both methods have 100% Coverage Rate on five experimental datasets; 2) the DMDI method identifies the significant clustering structure of Lenses and LA1 successfully, while the BKPlot having 0% CR on Lenses and LA1; 2) the DMDI method has lower False Discovery Rate than the BKPlot method on most of experimental datasets .

Table 2
Summary for seven small datasets

Datasets	#class	Method	Candidate ks	CR	FDR
T50I1000D200	{20}	DMDI	{20}	100%	0%
		BKPlot	{2,5,10,20}	100%	75%
T6I46D200	{5, 9}	DMDI	{5, 9}	100%	0%
		BKPlot	{3, 5, 9}	100%	33%
Lenses	{3}	DMDI	{3, 6, 12}	100%	66%
		BKPlot	{2, 4, 8, 16}	0%	100%
Soybean-small	{4}	DMDI	{2, 4, 7}	100%	66%
		BKPlot	{2, 4, 6}	100%	66%
Tr41	{10}	DMDI	{3, 5, 7, 10}	100%	75%
		BKPlot	{3, 6, 10}	100%	66%
Wap	{20}	DMDI	{10, 15, 20}	100%	66%
		BKPlot	{2, 5, 7, 9, 11, 16, 18, 20}	100%	87.5%
LA1	{6}	DMDI	{2, 4, 6, 8}	100%	75%
		BKPlot	{2, 5, 9}	0%	100%

We ran ACTD on dataset T50I1000D200 at the support 0.8 and the DMDI curve (Figure 13) clearly shows the candidate optimal number of T50I1000D200 is '20', which is same as the number of clustering structure we constructed. After converting the T50I1000D200 into a 200×1000 Boolean table for the ACE algorithm, we get the BKPlot result as shown in Figure 14. The BKPlot result includes the predefined '20' but having three more candidate cluster numbers than the result of DMDI curve. The DMDI curve (Figure 15) and BKPlot index graph (Figure 16) of T6I46D200 also show that both methods can identify the predefined number of clusters '5' and '9' . However, same as the one layer dataset, the BKPlot method has one more candidate number '3'. The results of synthetic data show that DMDI method is able to find all significant number of clusters with less noises than the BKPlot method.

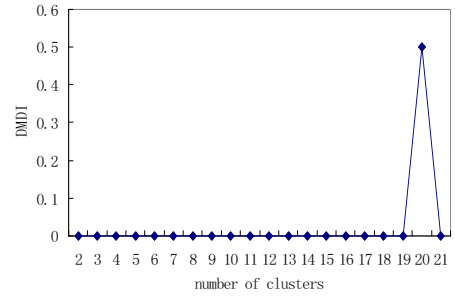

Fig. 13. DMDI curve of T50I1000D200 at Min-support=0.8

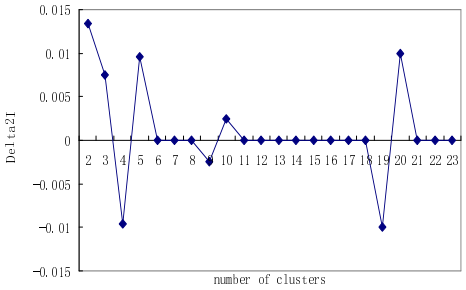

Fig. 14. BKPlot of T50I1000D200

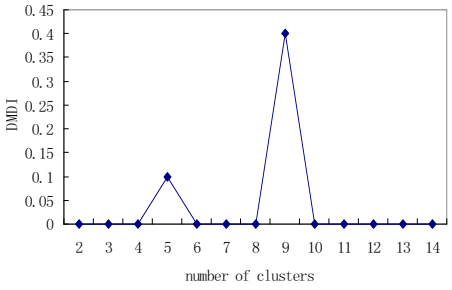

Fig. 15. DMDI curve of T6I46D200 at Min-support=0.8

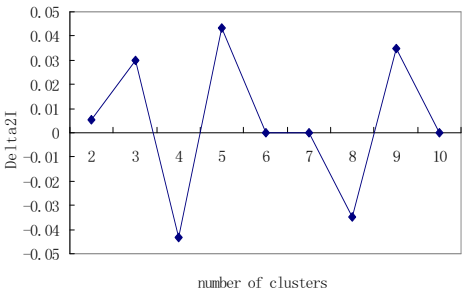

Fig. 16. BKPlot of T6I46D200

We did experiments on five real datasets and plotted their DMDI and BKPlot graphs respectively (Figure 17, 19, 18, 20, 21, 22, 23, 24, 25, 26). Again, Figure 17 and Figure 18 show that the DMDI result on transactional lenses is less noisy than the BKPlot's. The DMDI result ({3,6,12}) not only includes the predefined number of classes '3' but also has less candidate cluster numbers than that of BKPlot ({2, 4, 8, 16}). Figure 21 to Figure 26 show that DMDI method can discover the predefined classes for all three document datasets, while BKPlots fails to identify predefined class number of LA1. In addition, DMDI reduces the candidate best Ks from 8 given by BKPlot to 3 for Wap. However, the two methods perform equally on the dataset soybean-small(Figure 19 and Figure 20).

Here, the setting method of support values is discussed. The support value of algorithm ACTD is a real number between 0 and 1, i.e. (0, 1]. It decides the transactional cluster mode of each cluster during the whole agglomerative merging procedure. Essentially, the transactional cluster mode is a set of frequent items in a cluster and should best represent the feature of a cluster, thus the clustering structure can be found in high probability. Through the above experiments, we observe that the setting of support value is affected by the type of datasets, i.e. dense or sparse datasets. For dense datasets, the cluster modes obtained by the higher support value have less overlapping than the cluster modes obtained by the lower support value, which is ideal for detecting

clusters. While for sparse datasets, it is possible that the NULL cluster mode is outputted at high support value. In such situation, the appropriate cluster modes can be found at lower support value. So the support value 0.8 is appropriate for most datasets in our experiments, while the last three document datasets uses different support values according to their sparseness.

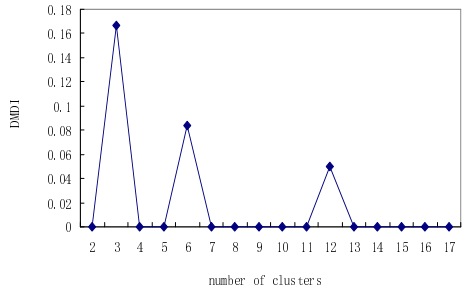

Fig. 17. DMDI curve of lenses at Min–support=0.8

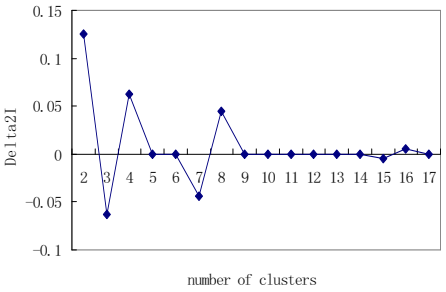

Fig. 18. BKPlot of lenses

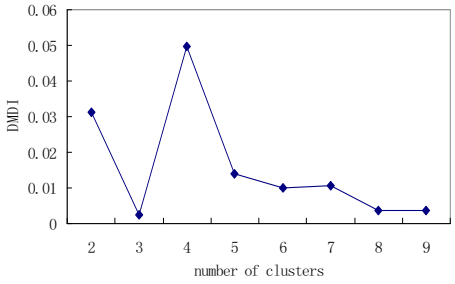

Fig. 19. DMDI curve of soybean-small at Min-support=0.8

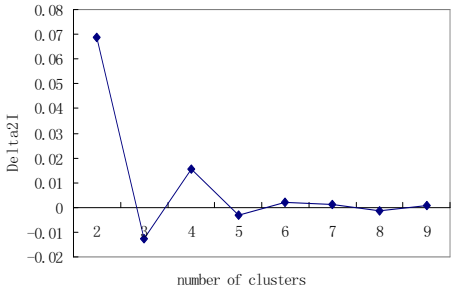

Fig. 20. BKPlot of soybean-small

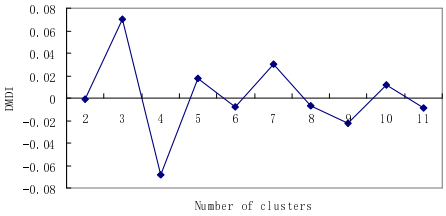

Fig. 21. DMDI curve of tr41 at Min–support=0.5

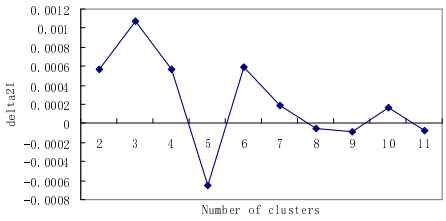

Fig. 22. BKPlot of Tr41

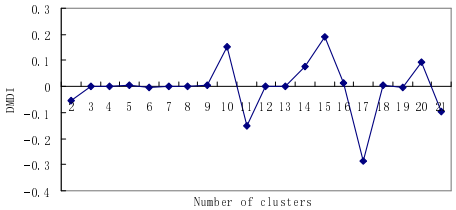

Fig. 23. DMDI curve of Wap at Min–support=0.6

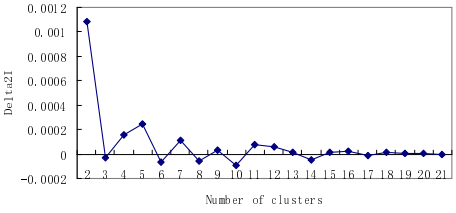

Fig. 24. BKPlot of Wap

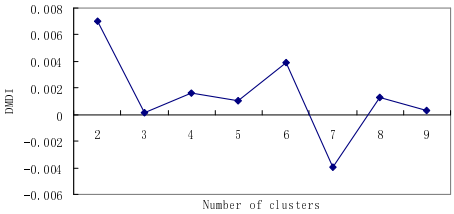

Fig. 25. DMDI curve of LA1 at Min–support=0.1

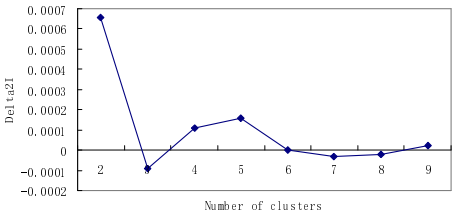

Fig. 26. BKPlot of LA1

4.3 Quality Evaluation on Sample Datasets

We have shown that ACTD is much faster than ACE for BKPlot on transactional datasets, which mean we can apply ACTD to a larger number of samples with the same time budget. In this section, we want to investigate if the sample size is fixed, what the quality difference is between these two methods, in terms of the coverage and noise rate of identifying the significant Ks. Similarly, Coverage Rate and False Discovery Rate are used to compare the quality of sample DMDI curves and sample BKPlot curves.

Two synthetic transactional datasets, which are constructed by ourselves and have predefined clustering structure, were used to test the capability of BKPlot and DMDI methods on dealing with large transactional datasets with the sampling method. The detail information of two synthetic datasets are given below.

T50I1000D1000 is a 5 times duplication dataset of T50I1000D200. The clustering structure of T50I1000D1000 is very clear and transactions within a cluster have same items. The predefined cluster number of T50I1000D1000 is 20 and each cluster has 50 transactions.

T20I500D20K is generated by a modified version of data generator in [2]. The T20I500D20K has predefined 10 clusters, and the coherence of transactions within a cluster is much looser than that of T50I1000D1000. The construction method of T20I500D20K is: 1)the 10 clusters are generated one by one and don't have any item overlapping between any two clusters;2) the generation of each cluster has different initial random seed; 3)While each cluster was generated, the following parameter values were set: average size of transaction = 20, number of items = 50, number of transactions =2000, average size of patterns = 4, number of patterns = 2000, correlation between consecutive patterns = 0.25.

Uniform sampling method was used to generate sample datasets from original datasets. We use sample sizes {200, 400, 800} on the dataset T50I1000D1000 and sample sizes {800, 1000, 2000} on the dataset T20I500D20K to generate

sample datasets. For each sample size, we generate 10 sample datasets. After running ACE and ACTD on 10 sample datasets, we average the values generated from 10 sample datasets for the sample BKPlot graph and the sample DMDI graph.

We summarize the results with two measures in Table 3. The Table 3 shows that both methods have 100% Coverage Rate, that is, two methods can indicate the inherent significant clustering structures correctly. However, the DMDI method has much lower False Discovery Rate than the BKPlot method on these sample datasets. Therefore, the DMDI method is same robust as the BKPlot method but less noisy candidate Ks. Next, we present these sample DMDI and BKPlot curves in detail.

Table 3
Summary of Sample DMDI and BKPlot Curves on two synthetic datasets

Datasets	Method	CR	FDR
T50I1000D1000 Samples	DMDI	100%	0%
	BKPlot	100%	82%
T20I500D20K Samples	DMDI	100%	25%
	BKPlot	100%	70%

For the synthetic dataset T50I1000D1000, which has clear clustering structure, the sample DMDI curves (Figure 27) merge very well and indicate the same cluster numbers as the original DMDI curves. Figure 28 shows the sample BKPlot curves have more or less deviations from original DMDI curves but have consistent peaks at $k = 20$, which indicate the size of merged clusters has influence on the peak values of BKPlot curves.

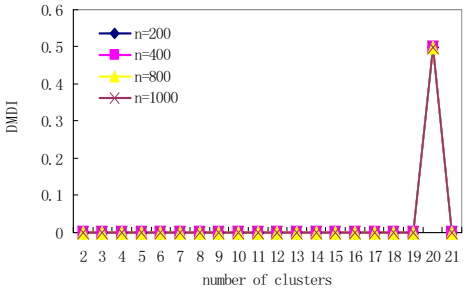

Fig. 27. sample DMDI curves of T50I1000D1000 at Min-support=0.8

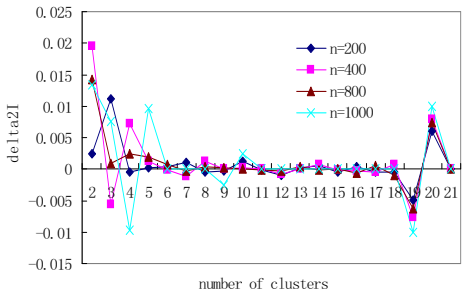

Fig. 28. sample BKPlot curves of T50I1000D1000

For the dataset T20I500D20K, the sample DMDI curves and sample BKPlot curves are shown in Figure 29 and Figure 30. These two figures show that both methods find the predefined cluster number 10 successfully and generate some noisy candidate cluster numbers. But the sample DMDI curves produce much less noisy candidate Ks than the sample BKPlot curves.

In addition to these sample DMDI and BKPlot curves, the average running time on these sample datasets are also presented in Figure 31 and Figure 32.

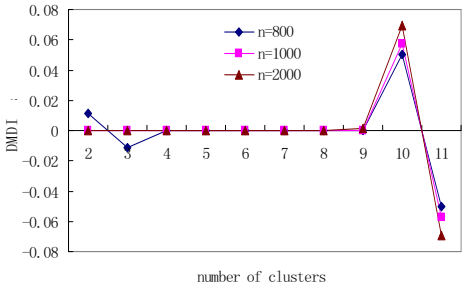

Fig. 29. Sample DMDI curves of T20I500D20K at Min-support=0.8

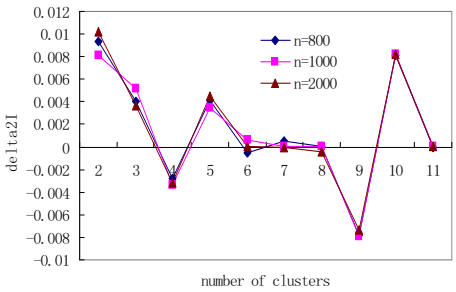

Fig. 30. Sample BKPlot curves of T20I500D20K

These time cost figures indicate that the ACTD algorithm is much faster than the ACE algorithm again.

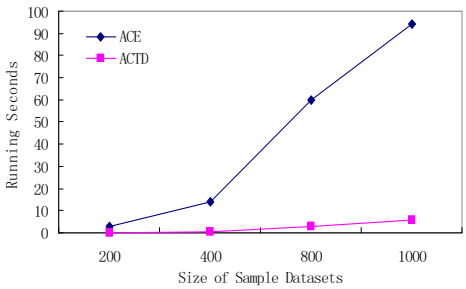

Fig. 31. Running time of ACTD and ACE on T50I1000D1K Sample datasets

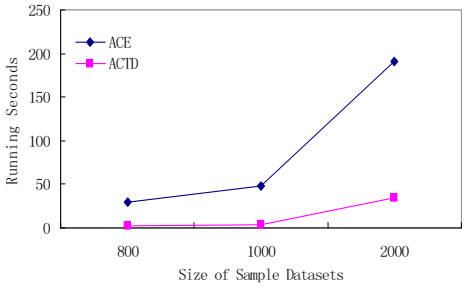

Fig. 32. Running time of ACTD and ACE on T20I500D20K Sample datasets

4.4 Capability of Dealing with Noise Transactions

In this set of experiments we test if the DMDI method can still find the clustering structures of datasets with noise transactions. The synthetic dataset T20I500D20K is used as the initial transactional dataset and various sizes of noise data are injected into the T20I500D20K uniformly to generate the mixed datasets. The noise rates are set as $0.5\%, 1\%, 5\%$ *and* 10% respectively in our experiments. For the T20I500D20K, the sizes of noise transactions are $\{100, 200, 1000, 2000\}$ corresponding to the above noise rates.

The way of generating noise transactions is described here. We first generate a noise dataset, then sample the noise dataset with various sizes to get noise transactions. The noise dataset is generated using same data generator [2] of T20I500D20K but different parameters. The detailed parameters are set as: average size of transaction = 4, number of items = 500, number of transactions =2000, average size of patterns = 4, number of patterns = 2000, correlation between consecutive patterns = 0.25. This way of generating noise dataset

makes the items of noise transactions overlapping with items of several clusters of T20I500D20K.

The noise dataset is sampled with size={100, 200, 1000, 2000} and the noise transactions are injected into the T20I500D20K uniformly to generate four mixed datasets T20I500D20k-N100, T20I500D20k-N200, T20I500D20k-N1000 and T20I500D20k-N2000. Then the four mixed datasets were uniformly sampled with size={800, 1000, 2000}. For each sample size, 10 sample datasets were generated. The DMDI graphs of these sample datasets (Figure 33 to Figure 36) show that DMDI method still can identify predefined classes correctly. We find the DMDI method is quite resilient to noise.

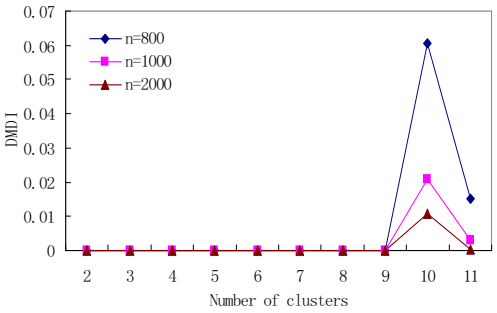

Fig. 33. Sample DMDI curves of T20I500D20K-N100 at Min-support=0.8

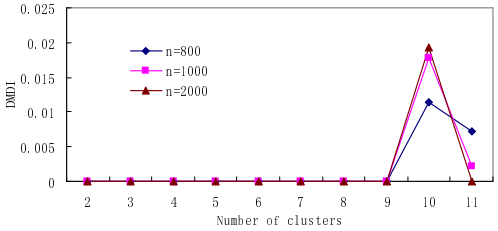

Fig. 34. Sample DMDI curves of T20I500D20K-N200 at Min-support=0.8

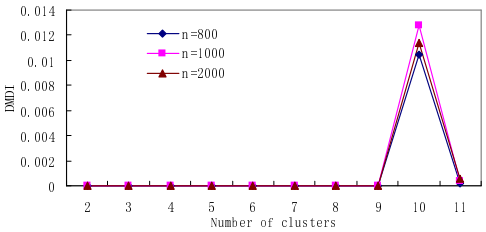

Fig. 35. Sample DMDI curves of T20I500D20K-N1000 at Min-support=0.8

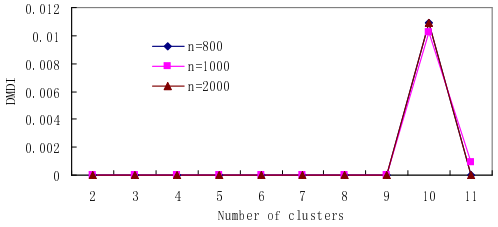

Fig. 36. Sample DMDI curves of T20I500D20K-N2000 at Min-support=0.8

4.5 Comparing DMDI with BIC

Among the classical clustering methods, the closest method to our proposed method on transactional data or categorical data is multinomial mixture modeling, the best K of which is often validated with the Bayesian Information Criterion (BIC) method [10]. In this Section, we compare our DMDI method with BIC to see if BIC curve can also effectively detect clustering structures in transactional datasets. For this purpose, we transform the transactional

data to Boolean data for multinomial mixture modeling. Below we give a brief introduction of BIC method.

The BIC method is a kind of model selection method. If the appropriate assumption is made about the prior distribution of the clusters, it can be used to select the optimal number of clusters. The model fitting is optimized by maximizing the likelihood of fitting the data to the mixture model. The generic form of BIC is then based on the maximum likelihood and the number of parameters used in estimation.

$$BIC = -2 \cdot \log likelihood + \log(n) \cdot \psi \tag{3}$$

where n is the number of sample records, and ψ is the number of parameters used in the modeling that include the number of clusters. Usually, the number of cluster corresponding to the minimum BIC is regarded as the optimal number of cluster.

We use AutoClass [4] to get BIC values for each K clusters in experiments. In AutoClass, ψ is specifically defined as $d(d+1)+K$, where d is the number of attributes. On the BIC curve, the best K happens at the minimum BIC. We run AutoClass on five small transactional datasets, i.e. T6I46D200, T50I1000D200, lenses, soybean-small and tr41, and the single_multinomial model, is used. The BIC Curves (Figure 37 to Figure 41) show: 1)The BIC method suggests K=4 for soybean-small, which is consistent with the predefined classes; 2)the BIC method suggests K=9 for two-layer dataset T6I46D200, which is consistent with the predefined classes. But it cannot find all the possible optimal number of clusters for multi-layer clustering structure; 3)The BIC suggestions for lenses, T50I1000D200 and tr41 are all not consistent with the predefined classes. Although BIC method can select the right number of clusters uniquely, it is not suitable for detecting multi-layer clustering structure. In addition, BIC method is not efficient on most of the experimental transactional datasets.

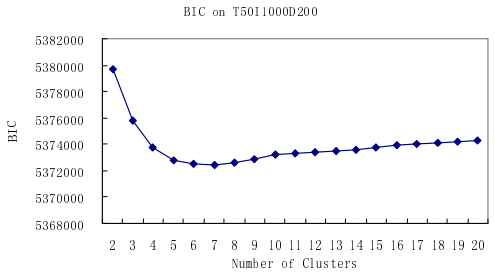

Fig. 37. BIC suggests $K = 7$ for T50I1000D200

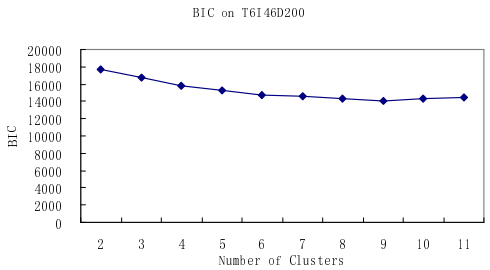

Fig. 38. BIC suggests $K = 9$ for T6I46D200

[4] http://ti.arc.nasa.gov/ic/projects/bayes-group/autoclass/

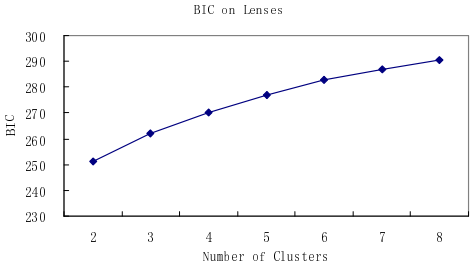

Fig. 39. BIC suggests $K = 2$ for lenses

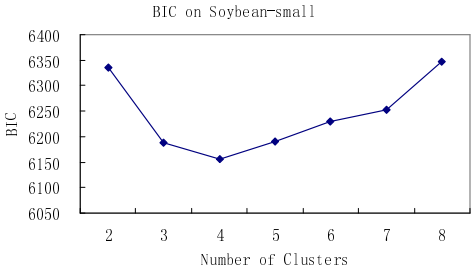

Fig. 40. BIC suggests $K = 4$ for soybean-small

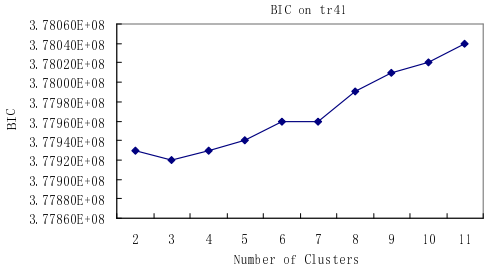

Fig. 41. BIC suggests $K = 3$ for tr41

4.6 Comparing SCALE with CLUTO

In our conference version of paper [**?**], we have compared our SCALE with a transactional clustering algorithm, i.e. CLOPE [20]. The experiments show the effectiveness of our SCALE. In this section, we compare our SCALE with CLUTO [5], a software package for clustering low and high dimensional document datasets. Below, we give a brief introduction of CLUTO.

For CLUTO, each document d is considered to be a vector in the term-space and the *tf-idf* term weighting model is used to generate document vector d, i.e. $d = (tf_1 log(n/df_1), tf_2 log(n/df_2), \ldots, tf_m log(n/df_m))$, where tf_i is the frequency of the *ith* term in the document and df_i is the number of documents that contain the *ith* term. The similarity of two documents d_i and d_j is $cos(d_i, d_j)$. Based on the cosine similarity, CLUTO developed several clustering criterion functions. The default one is $max \sum_{r=1}^{K} \sum_{d_i \in Sr} \cos(d_i, C_r)$, where C_r is the center vector of cluster S_r, i.e. average vector of all document vector of S_r. Since we are considering categorical data clustering, for fair comparison the frequency information is transformed to Boolean value before running CLUTO. Let the transformed Boolean feature be x_i. If $tf_i == 0$ then $x_i = 0$, otherwise $x_i = 1$.

CLUTO provides three different classes of clustering algorithms, which are based on the partitional, agglomerative, and graph-partitioning paradigms.

[5] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download

Concretely, six clustering methods are provided in CLUTO V2.1.1. Through combining clustering methods and criterion functions, users can get various clustering algorithms. In our experiments, we mainly compare our SCALE with five CLUTO clustering methods, i.e., c-direct, c-rb, c-rbr, c-agglo and c-bagglo, and the default clustering criterion is used.

The c-direct, c-rb and c-rbr use an approach inspired by the K-means algorithm to optimize the clustering criterion functions. They have initial and refinement phases. The initial phase has two ways to partition documents roughly, i.e. the repeated cluster bisection approach (c-rb and c-rbr) and direct K-way (c-direct) approach. For the bisection approach c-rb, a random pair of documents is used as the seeds of two clusters, and all documents are partitioned into two clusters. Then one of these clusters is selected and further bisected until the desired k clusters are found. The c-rbr is similar to the c-rb but at the end, the overall solution is globally optimized. For the direct K-way approach, random k documents are selected as cluster seeds. The remaining documents are assigned to the most similar seed. In the refinement phase, there are a number of iterations. During each iteration, the documents are visited in a random order and the document d_i is moved to the cluster that leads to the highest improvement of criterion value. The refinement phase ends if there is no document movement between clusters. To eliminate the sensitivity to the seed selection, the overall process is repeated a number of times, i.e. N different solutions, are computed. The default N is 10.

The c-agglo is the traditional agglomerative clustering algorithm. But the c-bagglo combines partitional and agglomerative methods. It introduces intermediate clusters obtained by partitional clustering algorithms to constrain the space over which agglomeration decisions are made.

Seven datasets are used in this group of tests, i.e. two synthetic datasets: T50I1000D200 and T6I46D200, two categorical datasets: Lenses and Zoo, three document datasets: Tr41, Wap and LA1. Zoo is from the UCI machine learning repository. It contains 101 data records for animals. Each data record has 18 attributes (animal name, 15 Boolean attributes, 1 numeric with set of values [0, 2, 4, 5, 6, 8] and animal type values 1 to 7) to describe the features of animals. The animal name and animal type values are ignored in our transformed file, while the animal type serves as an indication of domain-specific clustering structure. After transformed to transactional dataset, it has 36 items.

The **Purity** measure is used to evaluate the quality of clustering results in this group of tests. The purity of a clustering result is $P = \frac{1}{N}\sum_{i=1}^{K}\max(n_i^j)$, where $\max(n_i^j)$is the maximum number of transactions from the same class j in cluster i.

The number of desired clusters should be input before running CLUTO. In our experiments, the predefined class information is used as number of desired clusters when running CLUTO. For the SCALE, the cluster seeds at the predefined class number, which are generated by the clustering structure assessment step, form the initial clusters of clustering step.

The experimental results on seven datasets (Table 4) show: 1) On two well-structured synthetic transactional datasets, the clustering results of SCALE have the same purity as those of four CLUTO methods, i.e. c-rb, c-rbr, c-agglo and c-bagglo, and higher purity than that of c-direct method; 2) On two typical categorical datasets Lenses and Zoo, the SCALE obtains better clustering results than the five CLUTO methods; 3) On the three document datasets, the SCALE has better clustering result than CLUTO on WAP, and the purity of the SCALE on LA1 is higher than most of the CLUTO method except the c-rb, while the clustering result of SCALE on Tr41 is only better than that of the c-agglo. Overall, these results show that the transaction-specific SCALE prevails over CLUTO on most of the experimental datasets.

Table 4
Purity Summary of SCALE approach and five CLUTO methods

Datasets	#class	SCALE	c-direct	c-rb	c-rbr	c-agglo	c-bagglo
T50I1000D200	20	1.0	0.8	1.0	1.0	1.0	1.0
T6I46D200	9	1.0	0.9	1.0	1.0	1.0	1.0
Lenses	3	0.75	0.625	0.625	0.625	0.625	0.625
Zoo	7	0.93	0.842	0.89	0.86	0.83	0.73
Tr41	10	0.73	0.778	0.769	0.772	0.637	0.743
Wap	20	0.74	0.714	0.69	0.704	0.53	0.701
LA1	6	0.64	0.613	0.687	0.635	0.317	0.619

5 Conclusions

Although the problem of determining the optimal number of clusters is very challenging, we have shown that a coverage density-based method is promising for transactional datasets. This method is based on an intuitive transactional inter-cluster dissimilarity measure, Transactional-cluster-modes dissimilarity. Based on this measure, an agglomerative hierachical clustering algorithm is developed and its output, i.e. Merging Dissimilarity Indexes, are utilized to find the candidate number of clusters Ks. Our experimental results on both synthetic and real data showed that the proposed method is effective and highly efficient in finding the optimal number of clusters for transactional datasets.

References

[1] C. C. Aggarwal, C. Magdalena, and P. S. Yu. Finding localized associations in market basket data. *IEEE Trans. on Knowledge and Data Eng.*, 14(1):51–62, 2002.

[2] 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.

[3] A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[4] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. Limbo: Scalable clustering of categorical data. *Proc. of Intl. Conf. on Extending Database Technology (EDBT)*, pages 123–146, 2004.

[5] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. *Proc. of ACM SIGMOD Conference*, pages 49–60, 1999.

[6] D. Barbara, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)*, 2002.

[7] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. *Knowledge Discovery and Data Mining*, pages 254–260, 1999.

[8] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. *Proc. of ACM SIGKDD Conference*, 2004.

[9] C.-H. Chang and Z.-K. Ding. Categorical data visualization and clustering using subjective factors. *Data & Knowledge Engineering*, 53:243–262, June 2005.

[10] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): theory and results. pages 153–180, 1996.

[11] T. W. Darshit Parmar and J. Blackhurst. Mmr: An algorithm for clustering categorical data using rough set theory. *Data & Knowledge Engineering*, 63:879–893, December 2007.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[13] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus: Clustering categorical data using summaries. *Proceedings of Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 73–83, August 1999.

[14] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: an approach based on dynamical systems. *Proc. of Very Large Databases Conference (VLDB)*, 1998.

[15] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. *Proc. of ACM SIGMOD Conference*, 1998.

[16] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Proc. of IEEE Intl. Conf. on Data Eng. (ICDE)*, pages 512–521, March 1999.

[17] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster validity methods: Part I and II. *SIGMOD Record*, 31(2):40–45, 2002.

[18] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Workshop on Research Issues on Data Mining and Knowledge Discovery, (DMKD)*, 2(3):283–304, 1998.

[19] T. Li, S. Ma, and M. Ogihara. Entropy-based criterion in categorical clustering. *Proc. of Intl. Conf. on Machine Learning (ICML)*, 2004.

[20] Y. Yang, X. Guan, and J. You. Clope: A fast and effective clustering algorithm for transactional data. *Proc. of ACM SIGKDD Conference*, July 2002.