SCALE: A Scalable Framework for Efficiently Clustering Transactional Data

Hua Yan[1], Keke Chen*[2], Ling Liu[3], Zhang Yi[1] *

[1] Computational Intelligence Laboratory, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, P. R. China

[2] Department of Computer Science and Engineering, Wright State University, Dayton OH 45435, USA

[3]College of Computing, Georgia Institute of Technology, Atlanta,GA30280 USA

May 14, 2009

Abstract

This paper presents SCALE, a fully automated transactional clustering framework. The SCALE design highlights three unique features. First, we introduce the concept of Weighted Coverage Density as a categorical similarity measure for efficient clustering of transactional datasets. The concept of weighted coverage density is intuitive and it allows the weight of each item in a cluster to be changed dynamically according to the occurrences of items. Second, we develop the weighted coverage density measure based clustering algorithm, a fast, memory-efficient, and scalable clustering algorithm for analyzing transactional data. Third, we introduce two clustering validation metrics and show that these domain specific clustering evaluation metrics are critical to capture the transactional semantics in clustering analysis. Our SCALE framework combines the weighted coverage density measure for clustering over a sample dataset with self-configuring methods. These self-configuring methods can automatically tune the two important parameters of our clustering algorithms: (1) the candidates of the best number K of clusters; and (2) the application of two domain-specific cluster validity measures to find the best result from the set of clustering results.

*Contact author: Keke Chen
Department of Computer Science and Engineering, Wright State University, Dayton OH 45435, USA
email:keke.chen@wright.edu

We have conducted extensive experimental evaluation using both synthetic and real datasets and our results show that the weighted coverage density approach powered by the SCALE framework can efficiently generate high quality clustering results in a fully automated manner.

key words: transactional data clustering, cluster assessment, cluster validation, frequent itemset mining, weighted coverage density

1 Introduction

Data clustering is well-known as 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. It is widely recognized that numerical data clustering differs from categorical data clustering in terms of the types of data similarity measures used. Transactional data is a kind of special categorical data, and typical examples are market basket data, web usage data, customer profiles, patient symptoms records, and image features. Transactional data are generated by many applications from areas, such as retail industry, e-commerce, healthcare, CRM, and so forth. The volume of transactional data is usually large. Therefore, there are great demands for fast and yet high-quality algorithms for clustering large scale transactional datasets.

A transactional dataset consists of N *transactions*, each of which contains varying number of *items*. For example, $t_1 = \{milk, bread, beer\}$ and $t_2 = \{milk, bread\}$ are three-item transaction and two-item transaction respectively. A transactional dataset can be transformed to a traditional categorical dataset (a row-by-column Boolean table) by treating each item as an attribute and each transaction as a row. Although generic categorical clustering algorithms [5, 11, 6, 15, 14, 16, 20, 22] can be applied to the transformed Boolean dataset, the two key features of such transformed dataset: large volume and high dimensionality, make the existing algorithms inefficient to process the transformed data. For instance, a market basket dataset may contain millions of transactions and thousands of items, while each transaction usually contains about tens of items. The transformation to Boolean data increases the dimensionality from tens to thousands, which poses significant challenge to most existing categorical clustering algorithms in terms of efficiency and clustering quality.

Recently, a number of algorithms have been developed for clustering transactional data by utilizing specific features of transactional data, such as LargeItem [29], CLOPE [32], and CCCD [31]. However, all of the existing proposals suffer from one obvious drawback. All proposed clustering algorithms require users to

manually tune at least one or two parameters of the clustering algorithms in order to determine the number of clusters to be used by each run of the algorithm, and to find the best clustering result. For example, LargeItem [29] needs to set the support θ and the weight w, CLOPE [32] has a repulsion parameter r, and CCCD [31] has a parameter $MMCD$ as threshold on clusters merging. Unfortunately, the settings of all these parameters are manually executed and are different from dataset to dataset, making the tuning of these parameters extremely hard. No existing proposals, to the best of our knowledge, have offered general guideline for adequately setting and tuning these parameters.

In addition, there lacks of cluster validation methods to evaluate the quality of transactional clustering results. Because clustering is an unsupervised procedure, cluster validation is an important step to cluster analysis. Some generic measures or the interactive visualization method [10] have been developed for clustering numerical data based on statistical and geometrical properties [18]. Paper [24] proposes seeing clustering results as nodes in a graph, which opens the possibility of applying graph and lattice theory to compare clustering results. However, enumerating all the possible clusterings is a hard task for clustering large datasets. Due to the lack of meaningful pair-wise distance function, entropy-based measure has been widely used as a generic measure for categorical clustering [6, 22, 11]. However, such general metrics may not be effective as far as specific types of datasets are concerned, such as transactional data. It is recognized that meaningful domain-specific quality measures are more interesting [21, 18]. Surprisingly, very few of the existing transactional data clustering algorithms mentioned the clustering validation measure in terms of mining transactions.

In this paper we present a fast, memory-saving, and scalable clustering algorithm that can efficiently handle large transactional datasets without resorting to manual parameter settings. Our approach is based on two unique design ideas. First, we introduce the concept of Weighted Coverage Density (WCD) as intuitive categorical similarity measure for efficient clustering of transactional datasets. The WCD is an improved measure of Coverage Density (CD) [31] that uses the filled cell percentage on a 2D grid graph to measure the compactness of a group of data. The motivation of using weighted coverage density as our domain-specific clustering criterion is based on the observation that association rules mining over transactional data is inherently related to density-based data clustering [19]. Thus we define the weighted coverage density based on the concept of frequent itemsets [3]. Second, we develop two transactional-data-specific evaluation measures based on the concepts of large items [29] and coverage density respectively. Large Item Size Ratio (LISR) uses the percentage of large items in the clustering result to evaluate the clustering quality. Average pair-clusters Merging Index (AMI), applies coverage density to indicate the structural difference between clusters.

We design and develop a fully automated transactional clustering framework SCALE, and implement the

weighted coverage density measure based clustering algorithm and the two clustering validity metrics within SCALE. The SCALE framework performs the transactional data clustering in four steps, and can handle transactional datasets of small, medium, or large in size. In the first step it uses sampling to handle large transactional dataset, and then performs clustering structure assessment step to generate the candidate best number of clusters based on sample datasets. The clustering step uses the WCD measure based clustering algorithm to perform the initial cluster assignment and maximize the WCD measure through the iterative clustering refinement, until the WCD of the clustering result is not changed or the change is negligibly small. A small number of candidate clustering results are generated at the end of the clustering step. In the domain-specific evaluation step, we apply the two domain-specific measures (AMI and LISR) to evaluate the clustering quality of the candidate results produced and select the best one. We have conducted experimental evaluation using both synthetic and real datasets. Our results show that the weighted coverage density approach powered by the SCALE framework can generate high quality clustering results in an efficient and fully automated manner.

The rest of the paper is organized as follows. Section 2 gives an overview of the SCALE framework. Section 3 details the definitions of key concepts used in our clustering algorithm, the algorithm description and complexity analysis. The two measures AMI and LISR for clustering results evaluation are presented in Section 4. Our initial experimental evaluation results are reported in Section 5. We briefly introduce the related work in Section 6 and summarize our contributions in Section 7.

2 Overview of the SCALE Framework

We briefly describe the design of SCALE , a fully automated transactional clustering framework, before we discuss in detail the design and implementation of our weighted coverage density (WCD) measure based clustering algorithm and the two transactional-data-specific clustering validity metrics. The SCALE framework is designed to perform the transactional data clustering in four steps as shown in Figure 1.

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 cluster numbers based on sample datasets. In our prototype implementation, the

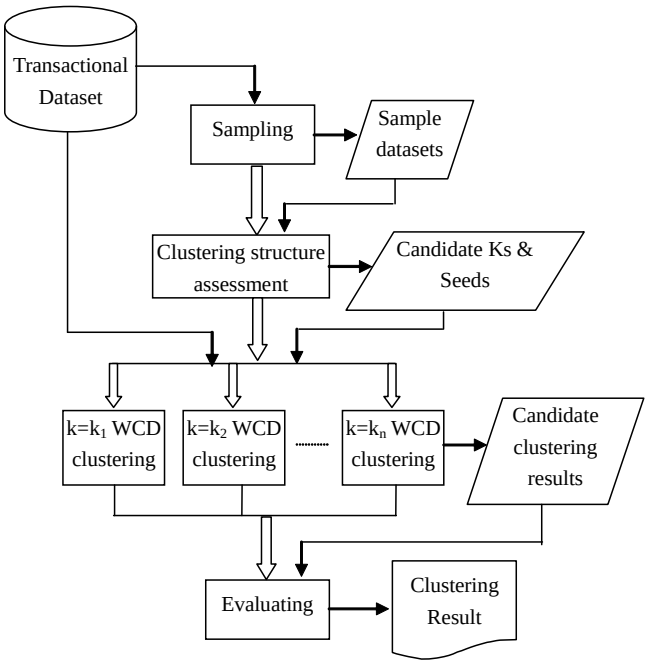

Figure 1: The SCALE framework

candidates of critical clustering structure are recommended by the Best K method BKPlot developed at Georgia Tech [11]. BKPlot method studies the entropy difference between the optimal clustering structures with varying K and reports only those Ks where the clustering structure changes dramatically between two neighboring optimal clustering schemes as the candidate best Ks, which greatly reduces the search on parameter space. In the SCALE prototype, we use a hierarchical algorithm proposed in [11] to generate high-quality approximate BKPlots, which can capture the candidate best Ks with small errors. The algorithm also generates 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 measure based clustering algorithm to perform initial cluster assignment. The initial assessment result is then used to guide the 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. Now 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.

3 WCD Clustering

In this section, we present the WCD measure based clustering algorithm for transactional data. The key design idea of the algorithm is the definition of the "Weighted Coverage Density" based clustering criterion, which tries to preserve as many frequent items as possible within clusters and controls the items overlapping between clusters implicitly.

We first introduce the concept of Coverage Density (CD) and Weighted Coverage Density (WCD) as intra-cluster similarity measures. The coverage density measure approximates the naive uniform item distribution in the clusters and is primarily used to describe the difference between item distributions, while the weighted coverage density measure describes the frequent-itemset preferred item distribution in the clusters and is used in clustering to preserve more frequent itemsets. We then compare CD and WCD based on their connection to statistical and information theoretic methods. Finally, we define the WCD measure based cluster criterion, and present an overview of the WCD measure based clustering algorithm, and a complexity analysis of the clustering algorithm. The WCD clustering algorithm takes a transactional dataset as its input and scans through the entire dataset and partitions transactions in an iterative process to maximize the overall *WCD* criterion.

3.1 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 $t_j = \{I_{j1}, I_{j2}, \ldots, I_{jl}\}$, $|t_j| = l$, such that $t_j \subseteq I$. A transaction clustering result C^K is a partition of D, denoted by $C_1, C_2, \ldots, C_K$, where $C_1 \bigcup \cdots \bigcup C_K = D, C_i \neq \phi, C_i \bigcap C_j = \phi, i \neq j$.

3.2 Intra-cluster Similarity Measures

In this section we illustrate the concept of Coverage Density (CD) and the concept of Weighted Coverage Density (WCD) as intra-cluster similarity measures. To provide an intuitive illustration of our development of these concepts, 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 that the item i is in the transaction j. For example, a simple transactional dataset {abcd, bcd, ac, de, def} can be visualized in Figure 2.

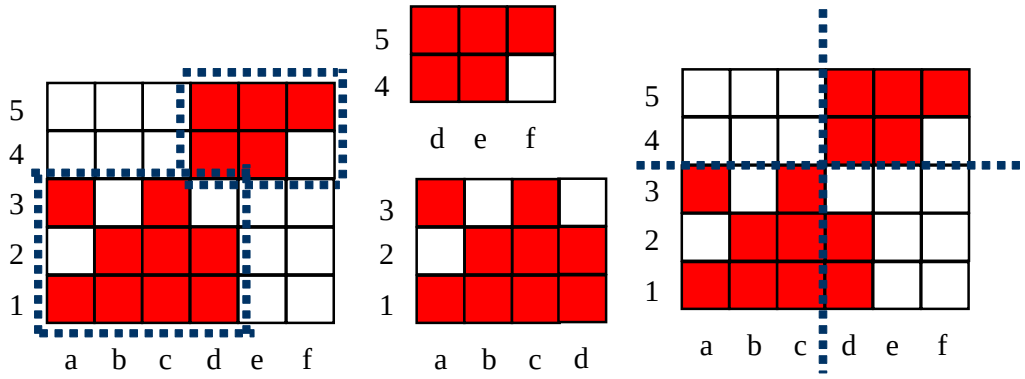

Figure 2: An example 2D grid graph

If we look at the filled area in the graph carefully, two naturally formed clusters appear, which are {abcd, bcd, ac} and {de, def} indicated by two rectangles in Figure 2. In the original graph there are 16 cells unfilled, but only 4 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 obtaining the minimized unfilled number of cells with appropriate number of partitions. Here, if we try to use bipartite graph based co-clustering method [1, 25, 13, 33, 12] to partition the transactions and the items, the result is shown by two straight lines in the right most part of Figure 2. Obviously co-clustering will result in the association loss between item c and item d. This is one of the reasons why we define our clustering problem as row clustering not as co-clustering in our application context. 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 our algorithm.

Bearing this intuition in mind, we define the first concept, *Coverage Density (CD)*, for evaluating the compactness of the partitions in terms of the unfilled cells only. In short, *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, the occurrence of item I_{kj} is $occur(I_{kj})$, and the sum occurrences of all items in cluster C_k is S_k, then the Coverage Density of cluster C_k is

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

Intuitively, the coverage density reflects the compactness of a cluster. Generally speaking, the larger the coverage density is, the higher the intra-cluster similarity among the transactions within a cluster.

However, the *CD* metric is insufficient to measure the density of frequent itemset, since in the *CD* metric each item has equal importance in a cluster. Namely, if the *cell (i, j)'s contribution* to the coverage density consists of *transactional contribution* T_i and the *item contribution* W_j. In *CD*, both transactional and item contributions are uniform, i.e., $T_i = T = \frac{1}{N_k}$ and $W_j = W = \frac{1}{M_k}$. *CD* can be represented as $T_i \times \sum_{j=1}^{M_k} occur(I_{kj}) \times W_j = T \times W \times \sum_{j=1}^{M_k} occur(I_{kj})$, treating each cell with the same importance as shown in Figure 4(a).

Another problem with the *CD* metric is the situation where two clusters may have the same *CD* but different filled-cell distributions. Consider the two clusters in Figure 3: is there any difference between the two clusters that have the same *CD* but different filled-cell distributions? This leads us to develop a heuristic rule for identifying and selecting a better distribution: we consider that a cluster with the coverage density focused on the high-frequency items is better in terms of compactness than a cluster with the same *CD* but the filled-cell distribution is somewhat *scattered* with respect to all items.

We now introduce the concept of *Weighted Coverage Density (WCD)* to serve for this purpose.

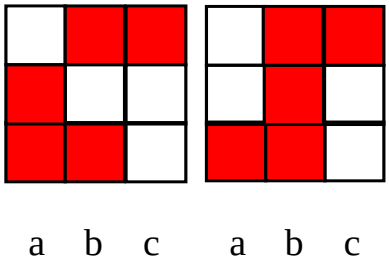

Figure 3: Two clusters with the same CD

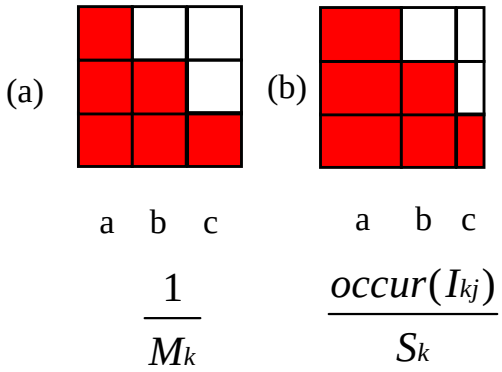

Figure 4: illustration of items contribution

Concretely, we put more "weight" on the items that have higher occurrence frequency in the cluster. This definition implies that the weight of each item is not a fixed one during the clustering procedure and it is changed along with the change of item distribution of a cluster. Thus, the item contribution W_j is no longer uniform as shown in Figure 4(b). Now, the item contribution W_j is defined as the ratio of occurrences of each item to the sum of occurrences of all items, namely,

$$W_j = \frac{occur(I_{kj})}{S_k}, \qquad st. \sum_{j=1}^{M_k} W_j = 1. \tag{2}$$

By Equation (2), and without changing the transactional contribution T, the *Weighted Coverage Density* of a cluster C_k can be defined as follows:

$$\begin{aligned} WCD(C_k) &= T \times \sum_{j=1}^{M_k} occur(I_{kj}) \times W_j \\ &= \frac{1}{N_k} \times \sum_{j=1}^{M_k} (occur(I_{kj}) \times \frac{occur(I_{kj})}{S_k}) \\ &= \frac{\sum_{j=1}^{M_k} occur(I_{kj})^2}{S_k \times N_k}. \end{aligned} \tag{3}$$

Recall the example in Figure 3, by Equation (3), the weighted coverage density of the cluster on the left is $\frac{9}{15}$, while the weighted coverage density of the cluster on the right is $\frac{11}{15}$. Therefore, the cluster on the right is better, which is consistent with our heuristic rule.

3.3 Comparing CD and WCD

In the above section we have given the intuitive definition of *CD* and *WCD*. We will show that *CD* and *WCD* are inherently connected to some important statistical concepts.

Let random variable X represent the frequency of any item, we consider that the occurrences of items in a cluster C_k follows a sample probability density function (PDF) of X, denoted by $f(X)$. It means that each value $occur(I_{kj})$, i.e., the frequency of item I_{kj}, is a sample of the random variable X. Therefore, *CD* and *WCD* are strongly related to the first moment (the expectation, $E[X] = \sum_{j=1}^{M_k} occur(I_{kj})/M_k$) and the second moment ($E[X^2] = \sum_{j=1}^{M_k} occur(I_{kj})^2/M_k$), i.e.,

$$CD(C_k) = T \times E[X],$$
$$WCD(C_k) = T \times \alpha \times E[X^2],$$

where $\alpha = M_k/S_k$ is a constant for the cluster C_k. Since $E[X^2] = E^2[X] + Var(X)$, for two clusters that have the same number of transactions (T), α, and $E[X]$, our clustering criterion of maximizing *WCD* will prefer the cluster having higher $Var(X)$, i.e., deviating more from the scenario that each item has similar frequency. We explain the implication of maximizing $Var(X)$ as follows. Let $p(X = I_{kj})$ be the probability of item I_{kj}

occurring in cluster C_k, i.e., $occur(I_{kj})/S_k$. Then, $-\sum_{j=1}^{M_k} p(X = I_{kj}) \log p(X = I_{kj})$ is the entropy of this cluster. For M_k number of items, this entropy is maximized when each item frequency $occur(I_{kj})$ is the same, i.e., $Var(X) = 0$, the variance of the item occurrences is minimized. However, reversely, we could not say that maximizing $Var(X)$ will directly lead to minimizing the entropy of item occurrences. Maximizing $Var(X)$ will certainly pull the entropy away from the maximum. On the other hand, although the entropy criterion has been shown effective in categorical data clustering in general [6, 22, 11], it is not easy to use the entropy of item frequencies as to use the WCD measure to intuitively interpret our goal of maximizing frequent itemsets. We will leave the comparison of WCD measure and the item frequency entropy measure in the future work.

3.4 Weighted Coverage Density based Clustering Criterion

We define the WCD-based clustering criterion in this section and outline the design of the WCD measure based clustering algorithm in the next section.

To define the WCD-based clustering criterion, we also take into account of the number of transactions in each cluster. For a clustering result $C^K = \{C_1, C_2, \ldots, C_K\}$, where $K < N$, we define the following *Expected Weighted Coverage Density (EWCD)* as our clustering criterion function.

$$\begin{aligned} EWCD(C^K) &= \sum_{k=1}^{K} \frac{N_k}{N} \times WCD(C_k) \\ &= \frac{1}{N} \sum_{k=1}^{K} \frac{\sum_{j=1}^{M_k} occur(I_{kj})^2}{S_k}. \end{aligned} \tag{4}$$

An EWCD-based clustering algorithm tries to maximize the *EWCD* criterion.

Let's give one example to show how the EWCD works. The transactional database is {abc, abcd, abce, bdfg, dgh, dgi} and its possible clustering schemes are {{abc, abcd, abce, bdfg}, {dgh, dgi}} and {{abc, abcd, abce}, {bdfg, dgh, dgi}}. Figure 5 shows the distribution of items in the original dataset and the two clustering schemes.

$E(1) = \frac{1}{6} \times (\frac{2^2+2^2+1^2+1^2}{6} + \frac{3^2+4^2+3^2+2^2+1^2+1^2+1^2}{15}) = 0.73333,$
$E(2) = \frac{1}{6} \times (\frac{1^2+3^2+1^2+3^2+1^2+1^2}{10} + \frac{3^2+3^2+3^2+1^2+1^2}{11}) = 0.80606.$
The result $E(1) < E(2)$ are consistent with the visual observation. First, the clustering scheme 2 has less unfilled cells than the scheme 1; second, transaction 4 has more overlapping with the frequent items in scheme

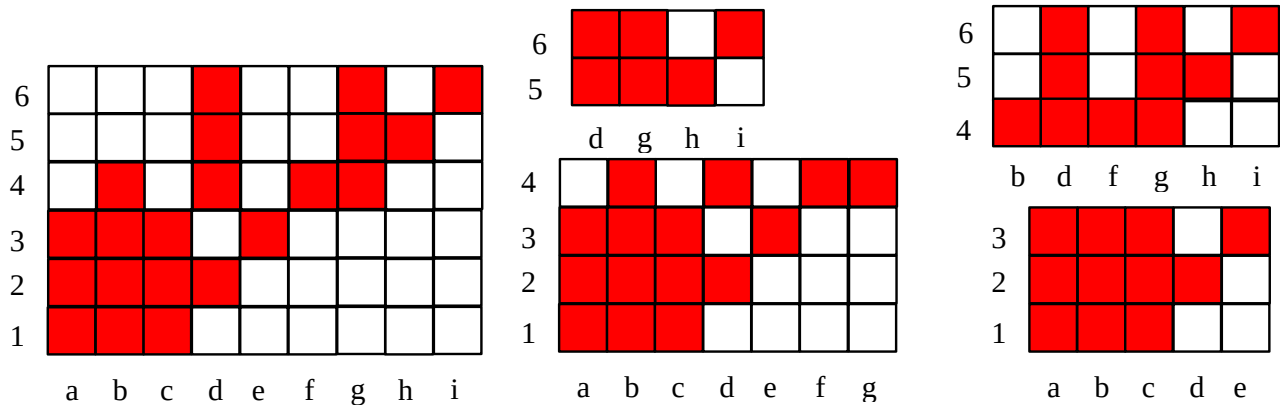

Figure 5: 2D grid graphs of original dataset and two clustering schemes

2 than in scheme 1.

However, if *EWCD* is used as the only metric in clustering procedure, an exception occurs when the number of clusters is not restricted - when every individual transaction is considered as a cluster, it will get the maximum *EWCD* over all clustering results. Therefore, the number of clusters needs to be either explicitly given or implicitly determined by other parameters. To avoid blindly setting k or tuning complicated parameters, we implement our clustering algorithm using the SCALE framework described in Section 2, where a set of candidate best number of clusters is suggested by the BKPlot method [11].

3.5 WCD Measure based Clustering Algorithm

The WCD measure based clustering algorithm uses a partition-based clustering approach. It scans the dataset iteratively to optimally assign each transaction to a cluster in terms of maximizing the EWCD criterion. The entire procedure of the WCD-based clustering can be divided into three phases: the clustering structure assessment phase, the WCD measure based initial cluster assignment, and the WCD measure based iterative clustering refinement phase. We call the first phase the WCD clustering preparation step, which can be performed by using an existing algorithm that can find the best K or best candidate Ks. While implementing the SCALE, the Best K method (BKPlot) we have developed [11] is used for finding the best candidate number of clusters. We refer to the second and third phases as the WCD clustering step, which is executed by the WCD measure based clustering algorithm.

In the initial cluster assignment phase, we take the outputs from the clustering structure assessment phase and produce an initial assignment using the WCD measure based clustering algorithm. Concretely, the clustering algorithm takes the K number of clusters and the cluster seeds at the best Ks as inputs to define the initial K

clusters. Each seed represents an initial cluster consisting of a few transactions. Given one of the best Ks, the WCD-based algorithm performs the clustering over the entire dataset. It reads the remaining transactions sequentially, and assigns each transaction to one of the K clusters, which maximizes the EWCD of the current clustering result. Our experiments show that the BKPlot method can efficiently help reduce the search space and get high quality clustering result.

Since the initial assignment produced in the second phase may not be optimal, in the iterative clustering refinement phase, the cluster assignment is refined in an iterative manner until no more improvement can be made with respect to WCD on the clustering result. Concretely, the algorithm reads each transaction in a randomly perturbed order, and check if the original cluster assignment is optimal in the sense that the EWCD metric is maximized. If it is not optimal, the transaction is moved to currently best fitted cluster, which increases the amount of EWCD the most. Any generated empty cluster is removed after a move. The iterative phase is stopped if no transaction is moved from one cluster to another in one pass for all transactions. Otherwise, a new pass begins. Note that the number of iterations may vary with respect to different random processing sequence and different clustering structure. It also depends on the number of clusters. According to our experience with the experiments, lambda is not controllable. For the well structured datasets, lambda is usually two or three. Lambda becomes larger for larger and noisy datasets.

A sketch of the pseudo code for the WCD measure based clustering algorithm is given in Algorithm 1.

Algorithm 1 WCD.main()

```
Input: Dataset file D of transactions; Number of clusters K; Initial K seeds
Output: K clusters
/*Phase 1 - Initialization*/
K seeds form the initial K clusters;
while not end of dataset file D do
    read one transaction t from D;
    add t into C_i that maximizes EWCD;
    write < t, i > back to D;
end while
/*Phase 2 - Iteration*/
while moveMark = true do
    moveMark = false;
    randomly generate the access sequence R;
    while has not checked all transactions do
        read < t, i >;
        if moving t to cluster C_j increases EWCD and i ≠ j then
            moveMark=true;
            write < t, j > back to D;
        end if
    end while
end while
```

Finding the destination cluster for each transaction is the key step in both phases, which needs to com-

pute/update EWCD for each possible assignment. To avoid unnecessary access and computation, the WCD clustering algorithm keeps the summary information of each cluster in main memory and updates it after each assignment. The summary information of cluster C_k includes the number of transactions N_k, the number of distinct items M_k, the sum occurrences of items S_k, the sum square occurrences of items $S_k^2 = \sum_{j=1}^{M_k} occur(I_{kj})^2$, the distinct items set I_k in cluster C_k, and the occurrences of each item $occur(I_{kj})$.

With the summary information, we are able to incrementally compute EWCD. Concretely, the two functions DeltaAdd and DeltaRemove can perform the incremental computing by adding one transaction into a cluster or removing one transaction from it respectively. Since the two functions are similar, we only provide outline of the function DeltaAdd in Algorithm 2. Let $t.I$ be the set of items in the transaction t.

Algorithm 2 WCD.deltaAdd(C_k, t)

```
float deltaAdd (C_k, t)
{
  S_k_new ⟵ S_k + |t|;
  ΔS_k^2 ⟵ 0;
  for (i = 0; i < |t|; i++) {
      if t.I[i] not exist in I_k then
      ΔS_k^2 ⟵ ΔS_k^2 + 1;
    else
      ΔS_k^2 ⟵ ΔS_k^2 + (occur(I_kj) + 1)^2 − occur(I_kj)^2;
  }
  return ((S_k^2 + ΔS_k^2)/S_k_new) − (S_k^2/S_k);
}
```

3.6 Complexity Analysis

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

The most time-consuming part is the update of EWCD to find the best cluster assignment which involves DeltaAdd and DeltaRemove. Since each DeltaAdd/DeltaRemove costs $O(|t|)$, the time complexity of the whole algorithm is $O(\lambda \times N \times K \times |t|)$, where λ is the number of iterations, N is the number of transactions in dataset, and $|t|$ is the average length of transaction. Usually λ, K and $|t|$ are much smaller than N, i.e. the running time is almost linear to the size of datasets. So the WCD measure based clustering algorithm is ideal for clustering very large transactional datasets.

4 Evaluating Clustering Results

Clustering is an unsupervised procedure trying to optimize certain objective function. This objective function could be domain-specific, highly complicated, and sometimes even not easy to specify. Typically, there are three scenarios. 1) In many cases, directly optimizing the objective function is often computationally intractable, and thus the clustering algorithms are all approximation algorithms. A typical example is entropy-based categorical clustering algorithms [11, 6, 22]. 2) Sometimes, it is even difficult to define an objective function, which covers multiple optimization goals. A commonly used approach to this scenario is to first generate the candidate clustering results with some generic clustering algorithms. The set of candidate results are then evaluated by a set of domain-specific measures and the optimal one is selected. 3) Most cases are between the above two. A design strategy has been made in our approach: instead of optimizing the multiple clustering criteria together, we optimize them in different stages. Concretely, we try to optimize the less costly measure in the costly clustering stage, which processes the entire large dataset, while other measures including the domain-specific measures are used later to select one clustering result from a set of candidate results.

We have shown that the criterion function *EWCD* can be directly optimized by the WCD measure based clustering algorithm. With the help of the BKPlot method, we can also determine a set of good candidates for the best number of clusters. In this section, we propose two quality measures, which are designed for the domain-specific cluster evaluation of transactional data. We use them to determine the optimal results from the candidate set.

LISR$-$ measuring the preservation of frequent itemsets

Since one of the popular applications of transactional data clustering is to find localized association rules [2], we propose a new measure called Large Item Size Ratio (*LISR*) to evaluate the percentage of Large Items [29] preserved by clustering. The more large items are preserved, the higher possibility the frequent itemsets are preserved in the clusters. An item is a "Large Item" when its occurrences in a cluster are above the user-specified proportion of transactions. We name the user-specified proportion as the minimum support τ, which is similar to the concept of "minimum support" in association rules mining. Let $IN(x)$ be the indicator function, i.e., if x is true, $IN(x) = 1$, otherwise, $IN(x) = 0$. The *LISR* computing formula is:

$$LISR(\tau) = \sum_{k=1}^{K} \frac{N_k}{N} \times \frac{\sum_{j=1}^{M_k} occur(I_{kj}) \times IN(occur(I_{kj}) \geq \tau \times N_k)}{S_k} \tag{5}$$

, where S_k stands for the total occurrences of all items in cluster C_k. In the above formula, the number of transactions in each cluster is taken into account in order to reduce the influence of noisy tiny clusters to the whole clustering result. A large *LISR* means high concurrences of items and implies the high possibility of finding more Frequent Itemsets [3] at the user-specified minimum support. In practice, users can provide different minimum supports they are interested for finding association rules, and then compare the *LISRs* of different clustering results to decide which clustering result is the most interesting one.

AMI$-$ measuring inter-dissimilarity of clusters

As we have shown previously, WCD measure evaluates the homogeneity of the cluster and tries to preserve more frequent itemsets, while CD only evaluates the homogeneity. Below we define a heuristic Clustering structural difference based on the CD measure, which is shown effective in describing the overall inter-cluster dissimilarity in experiments.

Given a pair of clusters C_i and C_j, the inter-cluster dissimilarity between the C_i and C_j is:

$$d(C_i, C_j) = \frac{N_i}{N_i + N_j} CD(C_i) + \frac{N_j}{N_i + N_j} CD(C_j) - CD(C_i \bigcup C_j). \tag{6}$$

The above Equation tries to compute the percentage of increased unfilled-cells in the new 2D mapping grid graph after merging two clusters.

Simplifying the above formula, we get $d(C_i, C_j) = \frac{1}{N_i+N_j}(S_i(\frac{1}{M_i} - \frac{1}{M_{ij}}) + S_j(\frac{1}{M_j} - \frac{1}{M_{ij}}))$, where M_{ij} is the number of distinct items after merging two clusters and thus $M_{ij} \geq \max\{M_i, M_j\}$. Because of $\frac{1}{M_{ij}} \leq \frac{1}{M_i}$ and$\frac{1}{M_{ij}} \leq \frac{1}{M_j}$, $d(C_i, C_j)$ is a real number between 0 and 1. Here, $S_i(\frac{1}{M_i} - \frac{1}{M_{ij}})$ describes the structural change caused by the cluster C_i. Not surprisingly, when two clusters have the same set of items, that is $M_i = M_j = M_{ij}$, $d(C_i, C_j)$ is zero. For two very different clusters having little overlapping between the sets of items, merging them will result in a large $d(C_i, C_j)$. Therefore, we say the above measure evaluates the structural difference between clusters. Two examples are given to illustrate the above situations in Figure 6 and Figure 7.

We propose the *Average pair-clusters Merging Index (AMI)* to evaluate the overall inter-dissimilarity of a clustering result having K clusters.

$$AMI = \frac{1}{K}\sum_{i=1}^{K} D_i,$$
$$D_i = Min\{d(C_i, C_j), i, j = 1, \ldots, K, i \neq j\}. \tag{7}$$

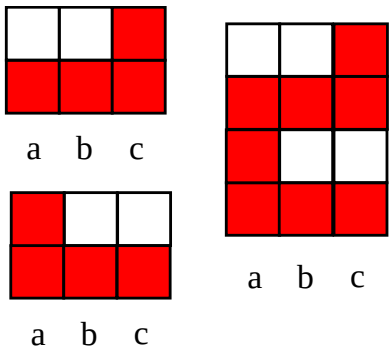

Figure 6: two clusters with zero dissimilarity

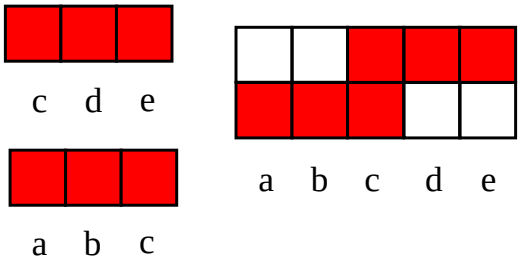

Figure 7: two clusters with positive dissimilarity

AMI is the average dissimilarity between all clusters. The larger the AMI is, the better the clustering quality.

Some traditional clustering methods try to optimize the clustering validity measure by combining intra-cluster similarity and inter-cluster dissimilarity [21, 18, 29]. However, this is extremely difficult in practice since we need some domain-specific weighting parameters to combine the intra-cluster similarity and the inter-cluster dissimilarity, and the setting of such parameters may differ from dataset to dataset. Thus, in our prototype implementation of the SCALE framework, we choose to optimize them separately: we optimize the intra-cluster EWCD measure with the WCD measure based clustering algorithm at the candidate best Ks, and use the AMI measure to select the best one. Our experiments show that AMI is an effective measure for indicating the globally distinctive clustering structure. In addition, LISR is used as another domain-specific measure in the SCALE framework. Our experiments in Section 5 show that the WCD clustering algorithm can generate high quality results reflecting the domain-specific structure.

5 Experiments

In this section we evaluate the SCALE framework using both synthetic and real datasets. The evaluation is compared with CLOPE [32], which is a clustering algorithm for transactional data without combining any other aided methods. The comparison is focused on the following three aspects to see if: 1)the SCALE clustering structure assessment step helps significantly reducing the time spent on parameter tuning; 2)the SCALE WCD clustering step is scalable to deal with large transactional data; 3)the SCALE evaluating step outputs high-quality clustering results in terms of the domain-specific measures.

Before reporting the results of our experiments, we first introduce the datesets used in the experiments.

5.1 Datasets

Our experiments have used two synthetic datasets: *Tc30a6r1000_2L* generated by us and *TxI4Dx Series* generated by synthetic data generator used in [3]. In addition, we used three real datasets: Zoo and Mushroom from the UCI machine learning repository [1] and Retail [8].

Tc30a6r1000_2L dataset is generated with a two-layer clustering structure that is clearly verifiable, as shown in Figure 8. We use the same method documented in [11] to generate the Tc30a6r1000_2L dataset. We want to use this synthetic dataset to test how well our WCD approach can perform when the critical clustering structures of the dataset are determined correctly at the clustering structure assessment step. It has 1000 records, and 30 columns, each of which has 6 possible attribute values. The top layer has 5 clusters with 200 data points in each cluster, four of which have two overlapping sub-clusters of 100 data points, respectively. In Figure 8, blank areas represent the same attribute value 0, while non-blank areas are filled with randomly generated attribute values ranging from 0 to 5. Since it is a generic categorical dataset, the attribute values are converted to items in order to run the WCD and CLOPE algorithms.

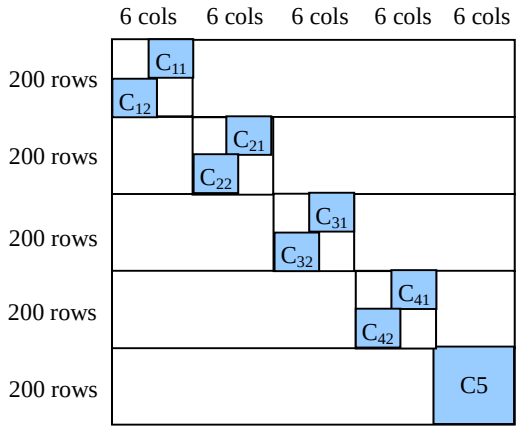

Figure 8: Structure of two-layer synthetic data

Zoo Real dataset Zoo from the UCI machine learning repository is used for testing the quality of clustering results. 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 also serves as an indication of domain-specific clustering structure.

Mushroom Real dataset Mushroom from the UCI machine learning repository contains 8124 instances, which

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

is also used for quality testing. Each data record has 22 categorical attributes (e.g. cap-shape, cap-color, habitat etc.) and is labeled either "edible" or "poisonous". The dataset contains 23 species of mushroom according to the literature. Therefore, we assume the domain-specific clustering structure could possibly have about 23 clusters. We use these knowledge to assess the clustering results and the effectiveness of the domain-specific measures.

Mushroom100k We also sample the mushroom data with duplicates to generate the mushroom100k of 100,000 instances as a real dataset for performance comparison with CLOPE.

TxI4Dx Series Data generator in [3] is used to generate large synthetic transactional datasets for performance test. We first give the symbols used in order to annotate the datasets. Three primary factors are the average transaction size T, the average size of the maximal potentially large itemsets I and the number of transactions D. For a dataset having $T = 10$, $I = 4$ and $100K$ transactions is denoted as $T10I4D100K$. The number of items and the number of maximal potentially large itemsets are always set to 1000 and 2000. We generate 5 groups of datasets from $T10I4D100K$ to $T10I4D500K$ by varying the number of transactions and each group has 10 randomly generated datasets at same parameters. We also generate 4 groups of datasets from $T5I4D100K$ to $T50I4D500K$ by setting the average length of transactions as 5, 10, 25 and 50. Also each group has 10 randomly generated datasets at same parameters.

Retail Retail [8] 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. It is used for performance evaluation of our approach on a real large transactional dataset with large quantity of items.

5.2 SCALE Clustering Structure Assessment

As mentioned before, all existing clustering algorithms require users to manually tune at least one or two parameters of the clustering algorithms in order to determine the number of clusters and to find the best clustering result. We call the algorithm running on each set of parameters as a trial clustering procedure and the corresponding clustering result as trial clustering result. It is not certain that how many trails we need in general to determine a close-to-optimal clustering result with a specific clustering algorithm. However, the number of trials can be definitely reduced if the clustering assessment can provide reliable information for reducing the search space over the parameter space. The SCALE framework takes this into consideration and uses our previously developed method BKPlot to significantly reduce the search space over the number of optimal clusters.

Let's take *Tc30a6r1000_2L*, *Zoo* and *mushroom* for example. After running ACE & BkPlots [11] on these datasets, the candidate Ks generated at the clustering structure assessment step are $\{3, 5, 9\}$ for Tc30a6r1000_2L, $\{2, 4, 7\}$ for Zoo and $\{2, 4, 6, 15, 19, 22, 29\}$ for Mushroom, which include the optimal domain-specific best Ks.

In contrast, the user of CLOPE will take much more efforts to find the optimal parameter setting. CLOPE has a repulsion parameter r, which implicitly controls the number of clusters. The r is a positive real number from zero to positive infinite. Since there is no guideline to find the appropriate setting of r, we usually start from setting r equal to a random number, such as $r = 1.0$, then do clustering and evaluate the clustering result to decide increasing or decreasing r in the next trial. Since there is no rule developed yet for determining the range of valid r and the possible step size, CLOPE will definitely take more trials before we are confident about the clustering result. For the dataset *Tc30a6r1000_2L*, we have to try 15 different r from 1.0 to 2.5 with step 0.1 to find the predefined cluster numbers $k = 5$ and $k = 9$. Some close-to-optimal results happen at $r = 2.0$ and $r = 2.4$, which have the domain-specific number of clusters. Similarly, if we start with $r = 1$, the close-to-optimal results happen at 2.4 for Zoo data, and 2.6 for mushroom data. Without the given hint of the number of domain-specific number of clusters, we will surely need more trials to determine the best clustering result. Therefore, the clustering assessment step in the SCALE framework can significantly improve the efficiency of the entire clustering process for transactional data.

5.3 Performance Study of WCD Clustering

We did performance study of WCD clustering in both synthetic datasets and real datasets to see if WCD clustering is scalable to deal with large transactional datasets. Since WCD clustering is quite memory saving, our experiments focused on the factors of time complexity. The time complexity of WCD measure based clustering algorithm is $O(\lambda \times N \times K \times |t|)$. Since the number of iterations, λ, is not controllable, we study the other three factors: the number of transactions N, the number of clusters K, and the average length of transactions $|t|$. The experiment results below show that the WCD clustering is scalable to large transactional data in terms of the related factors.

Performance Evaluation on TxI4Dx Series We first did experiments on 5 groups of T10I4Dx datasets with different size from 100K to 500K and set the number of clusters $K = 10, 50, 100$ separately. Each group has 10 randomly generated datasets with the same parameter setting and we average the running time of 10 datasets as the final running time of each group. Figure 9 and Figure 10 show that the overall cost is almost linear in

terms of the size of dataset and the number K of clusters.

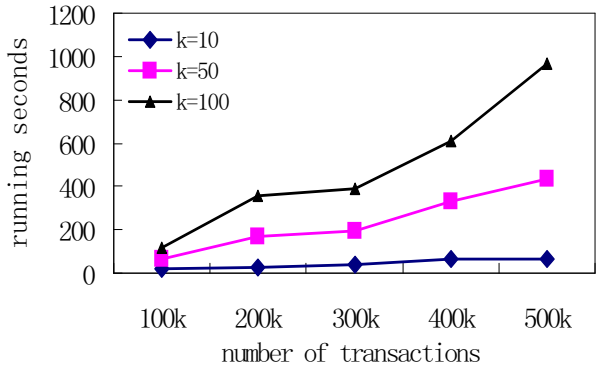

Figure 9: The total running time vs. the size of datasets on T10I4Dx datasets

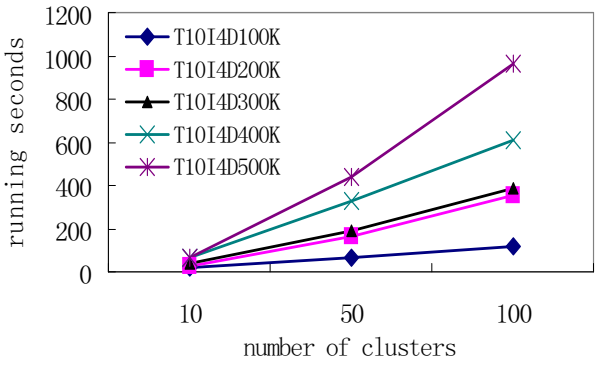

Figure 10: The total running time vs. the number k of clusters on T10I4Dx datasets

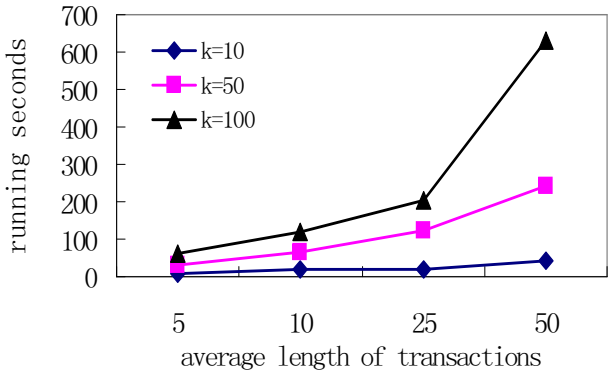

Figure 11: The total running time vs. the average length of transactions.

Then we did experiments on another 4 groups of TxI4D100K datasets with different average length of transactions and different number of clusters. Figure 11 shows for small number of clusters ($K \leq 50$), the average length of transactions is approximately linear to the running time, while for large K, such as $K = 100$, it becomes nonlinear. Since we are more interested in the clustering structure with small number of clusters ($K \leq 50$), the WCD measure based clustering algorithm is also scalable to the average length of transactions.

Performance Evaluation on Mushroom100k We compare the CLOPE and WCD on the running time of algorithms by varying the number of clusters and by varying the size of dataset.

First, we run CLOPE by varying r from 0.5 to 4.0 with step value 0.5. The running seconds and the number of clusters are reported for each r. The number of clusters is 17, 18, 27, 30, 31, 32, 41 and 64 for different r values, respectively. Correspondingly, we run WCD on these numbers of clusters and get WCD running seconds. The

comparison in Figure 12 shows that the cost of WCD is much less than that of CLOPE with respect to the number of clusters produced.

Second, we run CLOPE on 10%, 50% and 100% size of Mushroom100k with $r = 2.0$ and get the number K of clusters are 22, 23, and 30, respectively. Then we run WCD using the same numbers of clusters on the same set of datasets. The results in Figure 13 show that WCD is also much faster than CLOPE with respect to the size of the dataset.

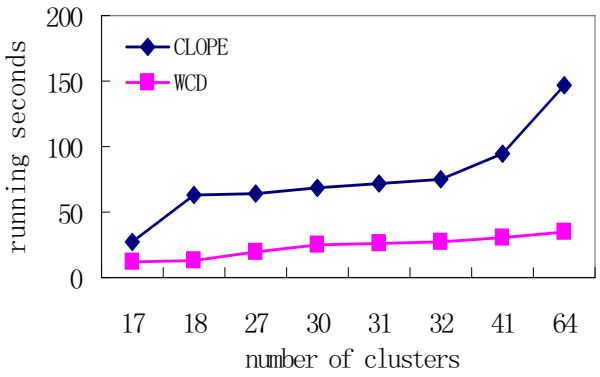

Figure 12: The total running time of CLOPE and WCD on Mushroom100k with varying K

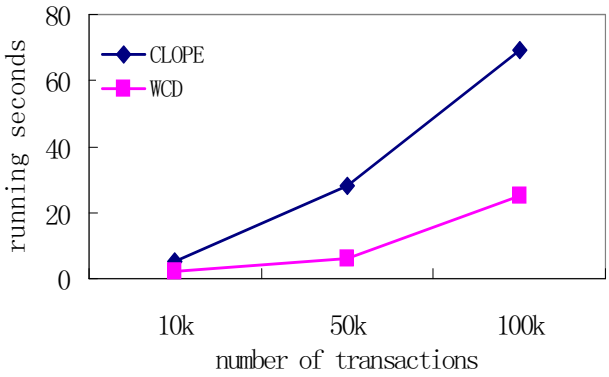

Figure 13: The total running time of CLOPE and WCD on Mushroom100k with varying size

Performance Evaluation on Retail Finally, we did experiments on Retail dataset by varying the number of

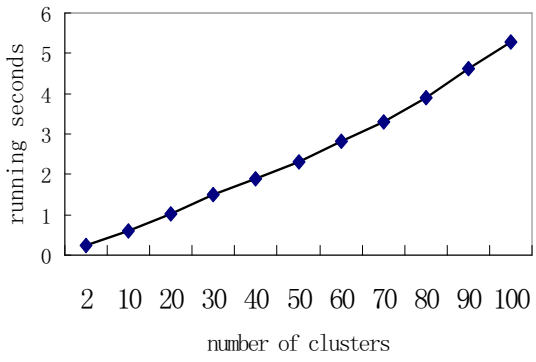

Figure 14: The average per-iteration time on Retail

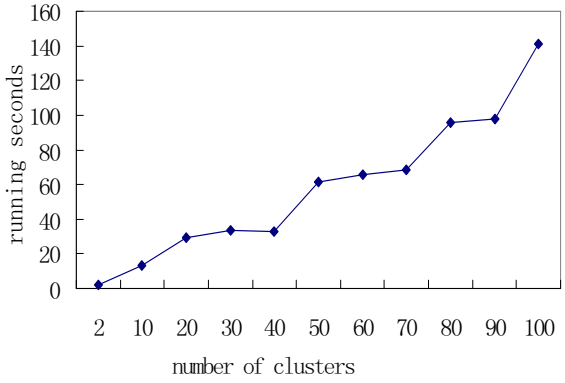

Figure 15: The total running time on Retail

clusters k from 2 to 100. The relationships between k and running time are shown in Figure 14 and Figure 15, which confirms the previous observations. Since the source code of CLOPE provided by authors cannot deal with variable length data, we couldn't compare our approach with CLOPE on Retail.

5.4 Quality of Clustering Results

In this group of experiments, we first use three measures: *LISR*, *AMI* and the Expected Entropy [6] to evaluate the quality of clustering results generated by SCALE and CLOPE. The result shows that the clustering quality of SCALE is higher than that of CLOPE in terms of the three measures. Apriori [4] is used to collect the number of frequent items preserved in the clustering results. We show that SCALE can preserve more frequent items in general.

We compare the LISR values in Figure 16 at various minimum support values from 0.6 to 1.0 in detail. (The larger the LISR value is, the better the result is in terms of preserving large items). LISR curves show that SCALE can preserve more large items than CLOPE especially on higher minimum support.

It is interesting to note that *AMI* index is consistent with the best Ks suggested by using BKPlot on some datasets. Figure 17 and 18 plot the *AMI* curves with varying K for datasets Tc30a6r100_2L and Zoo respectively. The global peak values (K=5 for Tc30a6r1000_2L and K=7 for Zoo), which indicate the clustering result with the highest inter-dissimilarity, are among the candidate Ks suggested by BKPlot. The best Ks are also identical to the predefined number of classes.

We summarize the best results of the two approaches in terms of LISR, AMI, and the classical Expected Entropy (EE) measure [6], in Table 1. For clear presentation, we briefly describe the definition of Expected Entropy. For a clustering result $C^K = \{C_1, C_2, \ldots, C_K\}$ of transaction database D, suppose N_i is the number of transaction of cluster C_i, M_i is the number of distinct item of cluster C_i, I_{ij} is the jth item in cluster C_i and $p(I_{ij})$ is the probability of item I_{ij}, the Expected Entropy of clustering result C^K is:

$$EE(C^K) = \sum_{i=1}^{K} \frac{N_i}{|D|} \times Entropy(C_i) = \sum_{i=1}^{K} \frac{N_i}{|D|} \times \sum_{j=1}^{M_i} (-p(I_{ij}) \log(p(I_{ij}))). \quad (8)$$

Table 1 shows that the SCALE clustering results have smaller Expected Entropies than that of CLOPE, which means the SCALE clustering results have higher intra-cluster similarities.

Table 1: LISR, AMI and Expected Entropy of SCALE and CLOPE clustering results

DataSets	approach	LISR(τ=0.9)	AMI	Expected Entropy
Tc30a6r1000_2L	SCALE	0.866867	0.161347	5.319538
	CLOPE	0.703273	0.094431	5.472092
Zoo	SCALE	0.704827	0.120252	4.281075
	CLOPE	0.53651	0.060841	4.399752
mushroom	SCALE	0.680278	0.120967	4.872727
	CLOPE	0.644397	0.105191	4.876047

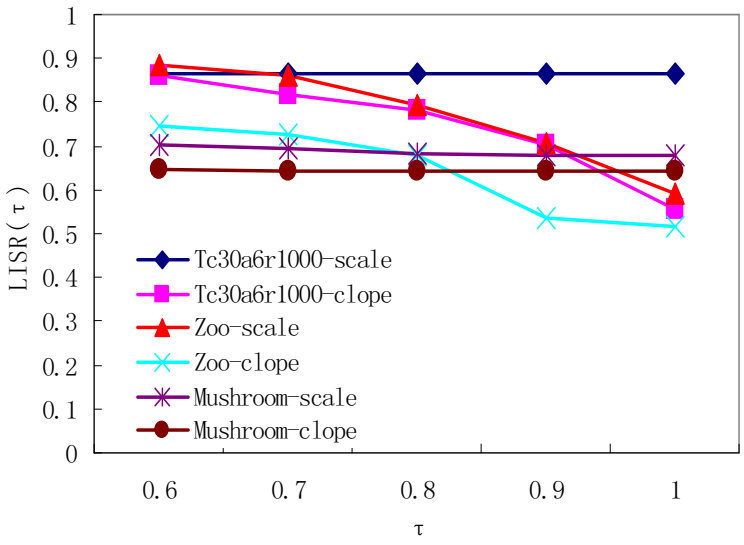

Figure 16: LISR graph

Since the *LISR* measures the preservation of frequent item sets indirectly, we use Apriori to find frequent itemsets in those clustering results and to see if the clustering results of SCALE have more frequent itemsets than that of CLOPE. We run frequent itemset mining on mushroom 2-cluster clustering results generated by SCALE and CLOPE.

Apriori is run on partitioned datasets at support 0.9%. With support higher than 1% we basically get no frequent itemsets, while with support $< 0.9\%$, there are too many frequent itemsets generated by both algorithms. For example, 228 frequent itemsets are found in SCALE partitioned database at support 0.7%, while 175 frequent itemsets are found in CLOPE partitioned database. The actual frequencies of many newly-found frequent itemsets at support 0.7% are very low. Figure 19 shows that much more frequent itemsets can be found in the WCD partitioned datasets than in CLOPE partitioned datasets at support 0.9%, which is consistent with the indication of *LISR* graph in Figure 16.

In summary, the experimental results above demonstrate that the LISR and AMI measures can help to find the clustering structures that are consistent with the documented structures. In general, the SCALE framework can generate better clustering results than CLOPE, with the additional advantage of no parameter tuning.

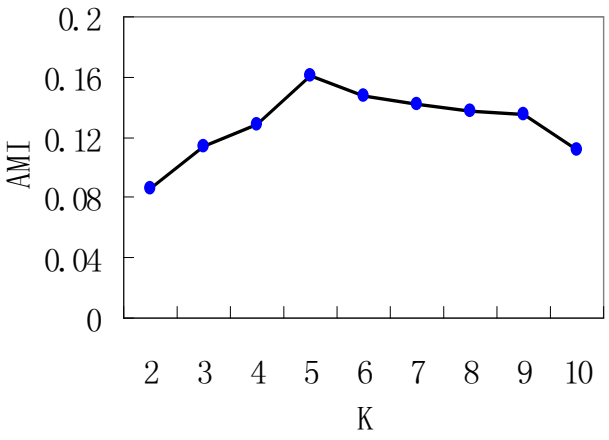

Figure 17: AMI curve for Tc30a6r1000_2L

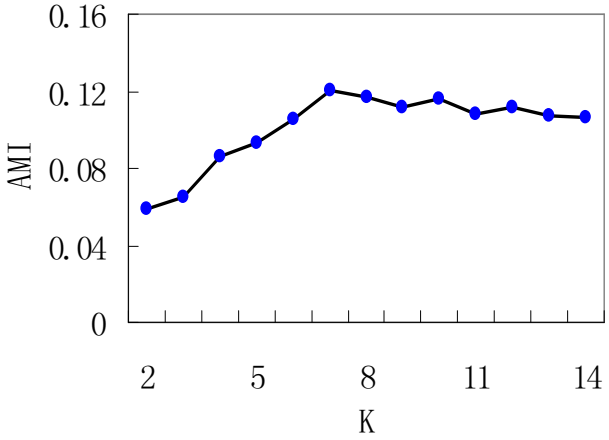

Figure 18: AMI curve for Zoo

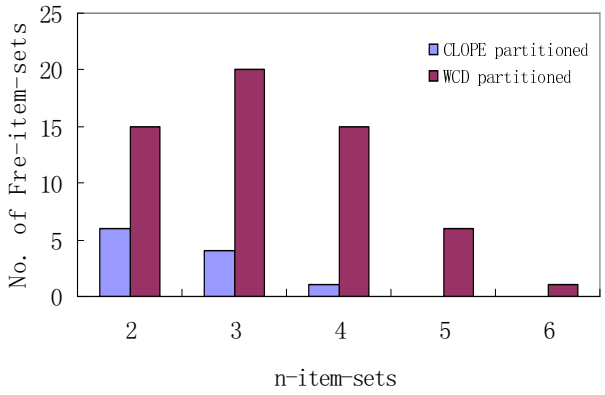

Figure 19: Frequent item-sets distribution at support 0.9%

6 Related Work

A number of algorithms have been developed for categorical data clustering in recent years [5, 11, 6, 15, 14, 16, 20, 22]. Some algorithms have studied distance-like pair-wise similarity measures, such as K-Modes [20] and ROCK [16]. While it is commonly recognized that a pair-wise similarity (e.g., cosine measure, the Dice and Jaccard coefficient, etc.) is not intuitive for categorical data, there have been algorithms using similarity measures for a set of records. The typical set-based similarity measures are based on information theory, such as expected-entropy in Coolcat [6], MC[22] and ACE[11], mutual information based similarity in LIMBO[5] and information bottleneck in [28], and minimum description length in Cross Association [9]. These algorithms have been focused on generic clustering structure of categorical data. However, only a few algorithms are dedicated to investigating the problems of transactional data clustering [32, 29, 31].

Another observation is that in the existing clustering algorithms, the number of clusters is determined either explicitly or implicitly through manually tuning of other parameter(s). For example, most data clustering methods, from the earliest k-modes [20] and ROCK [16] to the latest COOLCAT [6], LIMBO[5] and MC[22], take k as an input parameter of the algorithm. For algorithms using explicit parameter k, the user needs to assume the number of clusters at the beginning, which is extremely difficult in practice. Some other clustering algorithms, such as LargeItem[29], CLOPE[32] and CCCD [31], use one or two implicit input parameters to control the number of clusters. For implicit parameters, experimental results in [32, 29, 31] demonstrate the feasibility of using certain parameter setting to find the best K for a specific dataset through brute-force enumeration of various parameter settings, which is demonstrated by labeled datasets. Unfortunately, since real datasets for clustering do not have labels, it becomes extremely difficult to tune parameters and validate the clustering result for a real dataset. Therefore, it is practically important and necessary to develop and apply generic and domain-specific quality measures in cluster evaluation.

There are also some works on using bipartite graph theory to cluster transactional data [1, 25, 13, 33, 12]. Clustering algorithms based on partitioning bipartite graph usually generate co-clustering results, where columns and rows of the dataset are partitioned at the same time. If they are applied to transactional data, items and transactions are clustered simultaneously, which unnaturally splits the clusters that overlap over a few frequent items. Furthermore, the graph-based algorithms are often memory and time consuming, and inappropriate for clustering large transactional datasets.

In recent years, clustering data stream has been received extensive studies. Data stream has the following features. First, clustering computation works under the limited memory space. Second, the data can only be accessed one pass or limited passes. Third, the arrival of data must be in order. These features become the constraints on the design of data stream clustering algorithm. Most of existing data stream clustering algorithms are extended from the traditional clustering algorithms. For example, the classical K-Means is extended to clustering binary data stream [27]. Papers [17][7] improve the K-Median algorithm for clustering numerical data stream. SCLOPE [26] extends the CLOPE algorithm for clustering categorical data streams. For clustering transactional data stream, [23] incorporates an incremental clustering algorithm into different data stream model.

Finally, we would like to note that our initial results on the weighted coverage density clustering with small or medium datasets was reported in [30]. There have been several improvements since our earlier work. First, we have developed domain specific evaluation metrics to capture domain specific semantics in transactional clustering analysis. Second, we develop and implement the SCALE framework as a general architecture for

efficient clustering of transactional data of all sizes. Third, we have evaluated the SCALE framework and the WCD measure based clustering algorithm with the two new metrics with SCALE and demonstrate the effectiveness of our approach.

7 Conclusion

We have presented SCALE − a fully automated transactional data clustering framework, which eliminates the complicated parameter setting/tuning required by existing algorithms for clustering transactional data. Concretely, the SCALE framework is designed to perform the transactional data clustering in four consecutive steps. It uses sampling to handle large transactional dataset, and then performs clustering structure assessment step to generate the candidate "best Ks" based on sample datasets. The clustering step uses the WCD measure based clustering algorithm to perform the initial cluster assignment and the iterative clustering refinement. A small number of candidate clustering results are generated at the end of the clustering step. In the domain-specific evaluation step, the two domain-specific measures (AMI and LISR) are applied to evaluate the clustering quality of the candidate results produced and select the best one. Two unique features of SCALE are the WCD clustering algorithm − a fast, memory-saving and scalable method for clustering transactional data, and two transactional data specific cluster evaluation measures: LISR and AMI. We have reported our experimental evaluation results with both synthetic and real datasets. We show that compared to existing transactional data clustering methods, clustering under the SCALE framework can generate high quality clustering results in a fully automated manner with much higher efficiency for wider collections of transactional datasets.

There are some promising directions that can be explored in the future work. First, in our analysis, we have seen that the WCD measure is indirectly related to the entropy of item frequency. It would be interesting to perform some experimental comparison between the WCD measure and the entropy measure. Second, in the SCALE framework, we use the BKPlot method that was designed for categorical data clustering in general. We think if the characteristics of transactional data are fully explored and utilized as our WCD-based clustering algorithm does, we can design a better algorithm for determining the best K for transactional data clustering. Third, transactional data often arrives in the manner of streaming data. It would also be interesting to extend our work to handle transactional data streams.

Acknowledgments The first author is partly supported by Chinese 863 High-Tech Program under Grant 2008AA01Z132. The last author thanks for the partial support from grants in NSF CISE CyberTrust program, IBM SUR grant, and IBM Faculty Award.

References

[1] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. *Latin American Theoretical INformatics*, pages 598–612, 2002.

[2] 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.

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

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. of Very Large Databases Conference (VLDB)*, pages 487–499, 1994.

[5] 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.

[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)*, pages 582–589, 2002.

[7] B. Babcock, M. Datar, R. Motwani and L.O'Callaghan. Maintaining variance and k-medians over data stream windows. *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 234–243, 2003.

[8] 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.

[9] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. *Proc. of ACM SIGKDD Conference*, pages 79–88, 2004.

[10] K. Chen and L. Liu. VISTA: Validating and refining clusters via visualization. *Information Visualization*, 3(4):257–270, 2004.

[11] K. Chen and L. Liu. The "best k" for entropy-based categorical clustering. *Proc. of Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, pages 253–262, 2005.

[12] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[13] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. *Proceedings of ICDM 2001*, pages 107–114, 2001.

[14] 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.

[15] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: an approach based on dynamical systems. *Proc. of Very Large Databases Conference (VLDB)*, pages 311–322, 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] S. Guha, N. Mishra, and R. Motwani. Clustering Data Streams. *Proceeding of IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.

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

[19] T. Hastie, R. Tibshirani, and J. Friedmann. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[20] 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.

[21] A. K. Jain and R. C. Dubes. Data clustering: A review. *ACM Computing Surveys*, 31:264-323, 1999.

[22] T. Li, S. Ma, and M. Ogihara. Entropy-based criterion in categorical clustering. *Proc. of Intl. Conf. on Machine Learning (ICML)*, pages 68–75, 2004.

[23] Y. Li and R. Gopalan Clustering Transactional Data Streams. *Lecture Notes in Artificial Intelligence*, vol.4304, pages 1069-1073, 2006.

[24] M. Meilă. Comparing clusterings: an axiomatic view. *Proceedings of the 22nd international conference on Machine learning*, pages 577–584, 2005.

[25] N. Mishra, D. Ron, and R. Swaminathan. On finding large conjunctive clusters. *COLT*, pages 448–462, 2003.

[26] K.-l. Ong, W.y. Li, W.-k. Ng and E.-p. Lim. SCLOPE: An Algorithm for Clustering Data Streams of Categorical Attributes. *Proceedings of International Conference on Data Warehousing and Knowledge Discovery*, pages 209–218, 2004.

[27] C.Ordonez. Clustering Binary Data Streams with K-means. *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 10–17, 2003.

[28] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[29] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)*, pages 483–490, 1999.

[30] 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.

[31] H. Yan, L. Zhang, and Y. Zhang. Clustering categorical data using coverage density. *Proc. of Intl. Conf. on Advance Data Mining and Application*, pages 248–255, 2005.

[32] Y. Yang, X. Guan, and J. You. Clope: A fast and effective clustering algorithm for transactional data. *Proc. of ACM SIGKDD Conference*, pages 682–687, 2002.

[33] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Bipartite graph partitioning and data clustering. *CIKM*, pages 25–32, 2001.