

the same number of blocks ($H=1.6$, $H/W=0.32$). Clearly, clustering (1) is better since we prefer more overlapping among transactions in the same cluster.

From the above example, we can see that a larger height-to-width ratio of the histogram means better intra-cluster similarity. We apply this straightforward intuition as the basis of our clustering algorithm and define the global criterion function using the geometric properties of the cluster histograms. We call this new algorithm CLOPE - Clustering with sLOPE. While being quite effective, CLOPE is very fast and scalable when clustering large transactional databases with high dimensions, such as market-basket data and web server logs.

The rest of the paper is organized as follows. Section 2 analyzes the categorical clustering problem more formally and presents our criterion function. Section 3 details the CLOPE algorithm and its implementation issues. In Section 4, experiment results of CLOPE and LargeItem on real life datasets are compared. After some discussion of related works in Section 5, Section 6 concludes the paper.

2. CLUSTERING WITH SLOPE

Notations Throughout this paper, we use the following notations. A transactional database D is a set of transactions $\{t_1, \dots, t_n\}$. Each transaction is a set of items $\{i_1, \dots, i_m\}$. A clustering $\{C_1, \dots, C_k\}$ is a partition of $\{t_1, \dots, t_n\}$, that is, $C_1 \cup \dots \cup C_k = \{t_1, \dots, t_n\}$ and $C_i \neq \phi \wedge C_i \cap C_j = \phi$ for any $1 \leq i, j \leq k$. Each C_i is called a *cluster*. Unless otherwise stated, n , m , k are used respectively for the number of transactions, the number of items, and the number of clusters.

A good clustering should group together *similar* transactions. Most clustering algorithms define some criterion functions and optimize them, maximizing the intra-cluster similarity and the inter-cluster dissimilarity. The criterion function can be defined *locally* or *globally*. In the local way, the criterion function is built on the pair-wise similarity between transactions. This has been widely used for numerical data clustering, using pair-wise similarities like the L_p ($(\sum |x_i - y_i|^p)^{1/p}$) metric between two points. Common similarity measures for categorical data are the Jaccard coefficient ($|t_1 \cap t_2| / |t_1 \cup t_2|$), the Dice coefficient ($2 \times |t_1 \cap t_2| / (|t_1| + |t_2|)$), or simply the number of common items between two transactions [10]. However, for large databases, the computational costs of these local approaches are heavy, compared with the global approaches.

Pioneered by Wang et.al. in their LargeItem algorithm [13], global similarity measures can also be used in categorical data clustering. In global approaches, no pair-wise similarity measures between individual transactions are required. Clustering quality is measured in the cluster level, utilizing information like the sets of *large* and *small* items in the clustering. Since the computations of these global metrics are much faster than that of pair-wise similarities, global approaches are very efficient for the clustering of large categorical databases.

Compared with LargeItem, CLOPE uses a much simpler but effective global metric for transactional data clustering. A better clustering is reflected graphically as a higher height-to-width ratio.

Given a cluster C , we can find all the distinct items in the cluster, with their respective *occurrences*, that is, the number of transactions containing that item. We write $D(C)$ the set of distinct items, and $Occ(i, C)$ the occurrence of item i in cluster C . We can

then draw the *histogram* of a cluster C , with items as the X-axis, decreasingly ordered by their occurrences, and occurrence as the Y-axis. We define the *size* $S(C)$ and *width* $W(C)$ of a cluster C below:

$$S(C) = \sum_{i \in D(C)} Occ(i, C) = \sum_{t_i \in C} |t_i|$$

$$W(C) = |D(C)|$$

The *height* of a cluster is defined as $H(C) = S(C) / W(C)$. We will simply write S , W , and H for $S(C)$, $W(C)$, and $H(C)$ when C is not important or can be inferred from context.

To illustrate, we detailed the histogram of the last cluster in Figure 1 below. Please note that, geometrically in Figure 2, the histogram and the dashed rectangle with height H and width W have the same size S .

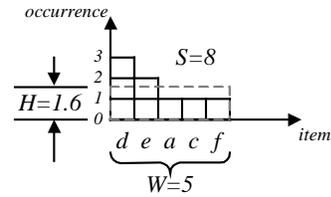


Figure 2. The detailed histogram of cluster $\{acd, de, def\}$.

It's straightforward that a larger height means a heavier overlap among the items in the cluster, and thus more similarity among the transactions in the cluster. In our running example, the height of $\{ab, abc, acd\}$ is 2, and the height of $\{acd, de, def\}$ is 1.6. We know that clustering (1) is better, since all the other characteristics of the two clusterings are the same.

However, to define our criterion function, height alone is not enough. Take a very simple database $\{abc, def\}$. There is no overlap in the two transactions, but the clustering $\{\{abc, def\}\}$ and the clustering $\{\{abc\}, \{def\}\}$ have the same height 1. Another choice works better for this example. We can use *gradient* $G(C) = H(C) / W(C) = S(C) / W(C)^2$ instead of $H(C)$ as the quality measure for cluster C . Now, the clustering $\{\{abc\}, \{def\}\}$ is better, since the gradients of the two clusters in it are all $1/3$, larger than $1/6$, the gradient of cluster $\{abc, def\}$.

To define the criterion function of a clustering, we need to take into account the shape of every cluster as well as the number of transactions in it. For a clustering $\mathbf{C} = \{C_1, \dots, C_k\}$, we use the following as a straightforward definition of the criterion function.

$$Profit(\mathbf{C}) = \frac{\sum_{i=1}^k G(C_i) \times |C_i|}{\sum_{i=1}^k |C_i|} = \frac{\sum_{i=1}^k \frac{S(C_i)}{W(C_i)^2} \times |C_i|}{\sum_{i=1}^k |C_i|}$$

In fact, the criterion function can be generalized using a parametric power r instead of 2 as follows.

$$Profit_r(\mathbf{C}) = \frac{\sum_{i=1}^k \frac{S(C_i)}{W(C_i)^r} \times |C_i|}{\sum_{i=1}^k |C_i|}$$

Here, r is a positive¹ real number called *repulsion*, used to control the level of intra-cluster similarity. When r is large, transactions within the same cluster must share a large portion of common items. Otherwise, separating these transactions into different clusters will result in a larger profit. For example, compare the two clustering for database $\{abc, abcd, bcde, cde\}$: (1) $\{\{abc, abcd, bcde, cde\}\}$ and (2) $\{\{abc, abcd\}, \{bcde, cde\}\}$. In order to achieve a larger profit for clustering (2), the profit for clustering

$$(2), \frac{\frac{7}{4^r} \times 2 + \frac{7}{4^r} \times 2}{4}, \text{ must be greater than that of (1), } \frac{\frac{14}{5^r} \times 4}{4}.$$

This means that a repulsion greater than $\ln(14/7)/\ln(5/4) \approx 3.106$ must be used.

On the contrary, small repulsion can be used to group sparse databases. Transactions sharing few common items may be put in the same cluster. For the database $\{abc, cde, fgh, hij\}$, a higher profit of clustering $\{\{abc, cde\}, \{fgh, hij\}\}$ than that of $\{\{abc\}, \{cde\}, \{fgh\}, \{hij\}\}$ needs a repulsion smaller than $\ln(6/3)/\ln(5/3) \approx 1.357$.

Now we state our problem of clustering transactional data below.

Problem definition Given D and r , find a clustering C that maximize $Profit_r(C)$.

```

/* Phrase 1 - Initialization */
1: while not end of the database file
2:   read the next transaction  $\langle t, \text{unknown} \rangle$ ;
3:   put  $t$  in an existing cluster or a new cluster  $C_i$ 
   that maximize profit;
4:   write  $\langle t, i \rangle$  back to database;

/* Phrase 2 - Iteration */
5: repeat
6:   rewind the database file;
7:   moved = false;
8:   while not end of the database file
9:     read  $\langle t, i \rangle$ ;
10:    move  $t$  to an existing cluster or new cluster  $C_j$ 
    that maximize profit;
11:    if  $C_i \neq C_j$  then
12:      write  $\langle t, j \rangle$ ;
13:      moved = true;
14: until not moved;
```

Figure 3. The sketch of the CLOPE algorithm.

3. IMPLEMENTATION

Like most partition-based clustering approaches, we approximate the best solution by iterative scanning of the database. However, as our criterion function is defined globally, only with easily computable metrics like size and width, the execution speed is much faster than the local ones.

Our implementation requires a first scan of the database to build the initial clustering, driven by the criterion function $Profit_r$. After

¹ In most of the cases, r should be greater than 1. Otherwise, two transactions sharing no common item can be put in the same cluster.

that, a few more scans are required to refine the clustering and optimize the criterion function. If no changes to the clustering are made in a previous scan, the algorithm will stop, with the final clustering as the output. The output is simply an integer label for every transaction, indicating the cluster id that the transaction belongs to. The sketch of the algorithm is shown in Figure 3.

RAM data structure In the limited RAM space, we keep only the current transaction and a small amount of information for each cluster. The information, called *cluster features*², includes the number of transactions N , the number of distinct items (or width) W , a hash of $\langle \text{item}, \text{occurrence} \rangle$ pairs occ , and a pre-computed integer S for fast access of the size of cluster. We write $C.occ[i]$ for the occurrence of item i in cluster C , etc.

Remark In fact, CLOPE is quite memory saving, even array representation of the occurrence data is practical for most transactional databases. The total memory required for item occurrences is approximately $M \times K \times 4$ bytes using array of 4-byte integers, where M is the number of dimensions, and K the number of clusters. Databases with up to 10k distinct items with a clustering of 1k clusters can be fit into a 40M RAM.

The computation of profit It is easy to update the cluster feature data when adding or removing a transaction. The computation of profit through cluster features is also straightforward, using S , W , and N of every cluster. The most time-sensitive parts in the algorithm (statement 3 and 10 in Figure 3.) are the comparison of different profits of adding a transaction to all the clusters (including an empty one). Although computing the profit requires summing up values from all the clusters, we can use the value change of the current cluster being tested to achieve the same but much faster judgement.

```

1: int DeltaAdd(C, t, r) {
2:   S_new = C.S + t.ItemCount;
3:   W_new = C.W;
4:   for (i = 0; i < t.ItemCount; i++)
5:     if (C.occ[t.items[i]] == 0) ++W_new;
6:   return S_new * (C.N + 1) / (W_new)^r - C.S * C.N / (C.W)^r;
7: }
```

Figure 4. Computing the delta value of adding t to C .

We use the function $DeltaAdd(C, t, r)$ in Figure 4 to compute the change of value $\frac{C.S \times C.N}{(C.W)^r}$ after adding transaction t to cluster C .

The following theorem guarantees the correctness of our implementation.

Theorem If $DeltaAdd(C_i, t)$ is the maximum, then putting t to C_i will maximize $Profit_r$.

Proof: Observing the profit function, we find that the profits of putting t to different clusters only differ in the numerator part of the formula. Assume that the numerator of the clustering profit before adding t is X . Subtracting the constant X from these new numerators, we get exactly the values returned by the $DeltaAdd$ function.

Time and space complexity From Figure 4, we know that the time complexity of $DeltaAdd$ is $O(t.ItemCount)$. Suppose the

² Named after BIRCH [14].

average length of a transaction is A , the total number of transactions is N , and the maximum number of clusters is K , the time complexity for one iteration is $O(N \times K \times A)$, indicating that the execution speed of CLOPE is affected linearly by the number of clusters, and the I/O cost is linear to the database size. Since only one transaction is kept in memory at any time, the space requirement for CLOPE is approximately the memory size of the cluster features. It is linear to the number of dimensions M times the maximum number of clusters K . For most transactional databases, it is not a heavy requirement.

4. EXPERIMENTS

In this section, we analyze the effectiveness and execution speed of CLOPE with two real-life datasets. For effectiveness, we compare the clustering quality of CLOPE on a labeled dataset (mushroom from the UCI data mining repository) with those of LargeItem [13] and ROCK [7]. For execution speed, we compare CLOPE with LargeItem on a large web log dataset. All the experiments in this Section are carried out on a PIII 450M Linux machine with 128M memory.

4.1 Mushroom

The mushroom dataset from the UCI machine learning repository (<http://www.ics.uci.edu/~mllearn/MLRepository.html>) has been used by both ROCK and LargeItem for effectiveness tests. It contains 8,124 records with two classes, 4,208 edible mushrooms and 3,916 poisonous mushrooms. By treating the value of each attribute as items of transactions, we converted all the 22 categorical attributes to transactions with 116 distinct items (distinct attribute values). 2480 missing values for the stalk-root attribute are ignored in the transactions.

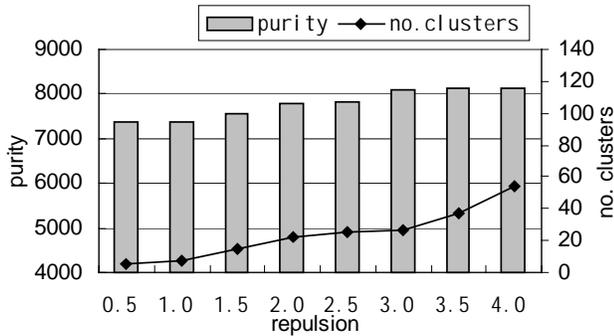


Figure 5. The result of CLOPE on mushroom.

We try different repulsion value from 0.5 to 4, with a step of 0.1. A few of the results are shown in Figure 5.

To make a general impression of the clustering quality, we use two metrics in the chart. The *purity* metric is computed by summing up the larger one of the number of edibles and the number of poisonous in every cluster. It has a maximum of 8124, the total number of transactions. The *number of clusters* should be as few as possible, since a clustering with each transaction as a cluster will surely achieve a maximum purity.

When $r=2.6$, the number of clusters is 27, and there is only one clusters with mixed records: 32 poisonous and 48 edibles (*purity*=8092). When r reaches 3.1, there are 30 clusters with perfect classification (*purity*=8124). Most of these results require

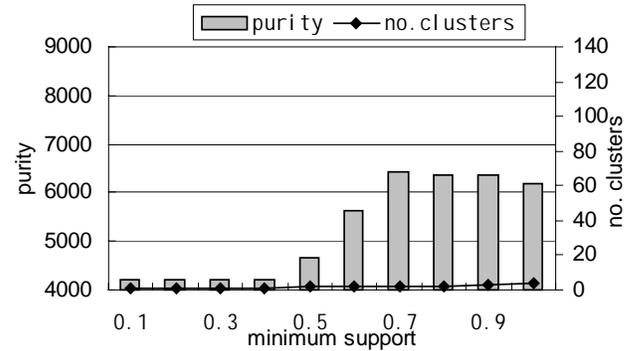
at most 3 scans of the database. The number of transactions in these clusters varies, from 1 to 1726 when $r=2.6$.

The above results are quite close to results presented in the ROCK paper [7], where the only result given is 21 clusters with only one impure cluster with 72 poisonous and 32 edibles (*purity*=8092), by a *support* of 0.8. Consider the quadratic time and space complexity of ROCK, the results of CLOPE are quite appealing.

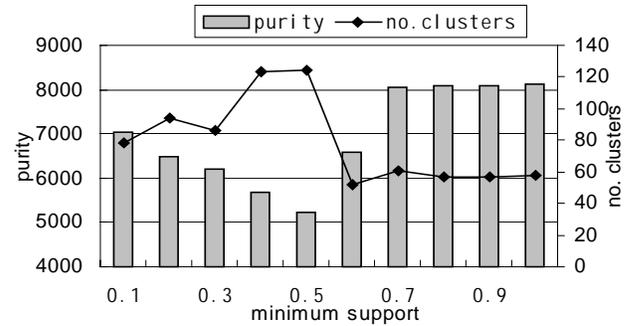
The results of LargeItem presented in [13] on the mushroom dataset were derived hierarchically by recursive clustering of impure clusters, and are not comparable directly. We try our LargeItem implementation to get the direct result. The criterion function of LargeItem is defined as [13]:

$$Cost_{\theta,w}(\mathbf{C}) = w \times Intra + Inter$$

Here θ is the minimum support in percentage for an item to be large in a cluster. *Intra* is number of distinct small (non-large) items among all clusters, and *Inter* the number of overlapping large items, which equals to the total number of large items minus the distinct number of large items, among all clusters. A weight w is introduced to control the different importance of *Intra* and *Inter*. The LargeItem algorithm tries to minimize the cost during the iterations. In our experiment, when a default $w=1$ was used, no good clustering was found with different θ from 0.1 to 1.0 (Figure 6(a)). After analyzing the results, we found that there was always a maximum value for *Intra*, for all the results. We increased w to make a larger *Intra* more expensive. When w reached 10, we found pure results with 58 clusters at support 1. The result of $w=10$ is shown in Figure 6(b).



(a) weight for *Intra* = 1



(b) weight for *Intra*=10

Figure 6. The result of LargeItem on mushroom.

Our experiment results on the mushroom dataset show that with very simple intuition and linear complexity, CLOPE is quite effective. The result of CLOPE on mushroom is better than that of LargeItem and close to that of ROCK, which has quadratic complexity to the number of transactions. The comparison with LargeItem also shows that the simple idea behind CLOPE works quite well even without any explicit constraint on inter-cluster dissimilarity.

Sensitivity to data order We also perform sensitivity test of CLOPE on the order of input data using mushroom. The result in Figure 5 and 6 are all derived with the original data order. We test CLOPE with randomly ordered mushroom data. The results are different but very close to the original ones, with a best result of reaching purity=8124 with 28 clusters, at $r=2.9$, and a worst result of reaching purity=8124 with 45 clusters, at $r=3.9$. It shows that CLOPE is not very sensitive to the order of input data. However, our experiment results on randomly ordered mushroom data show that LargeItem is more sensitive to data order than CLOPE.

4.2 Berkeley web logs

Apart from market basket data, web log data is another typical category of transactional databases. We choose the web log files from <http://www.cs.berkeley.edu/logs/> as the dataset for our second experiment and test the scalability as well as performance of CLOPE. We use the web logs of November 2001 and preprocess it with methods proposed in [3]. There are about 7 million entries in the raw log file and 2 million of them are kept after non-html³ entries removed. Among these 2 million entries, there are a total of 93,665 distinct pages. The only available *client IP* field is used for user identification. With a session idle time of 15 minutes, 613,555 sessions are identified. The average session length is 3.34.

For scalability test, we set the maximum number of clusters to 100 and run CLOPE ($r=1.0, 1.5, 2.0$) and LargeItem ($\theta = 0.2, 0.6, 1$, with $w=1$) on 10%, 50% and 100% of the sessions respectively. The average per-iteration running time is shown in Figure 7.

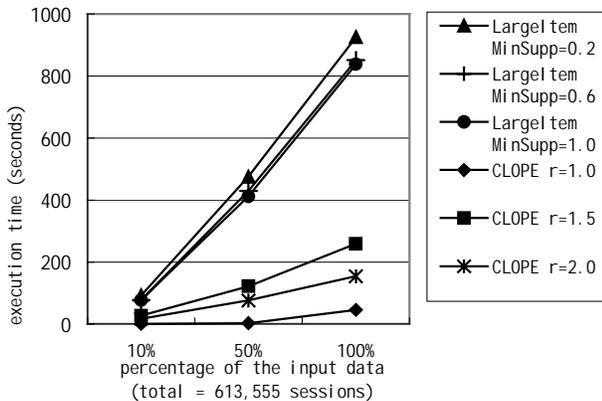


Figure 7. The running time of CLOPE and LargeItem on the Berkeley web log data.

From Figure 7, we can see that the execution time of both CLOPE and LargeItem are linear to the database size. For non-integer repulsion values, CLOPE runs slower for the float point

computational overhead. All these results reach the maximum number of clusters allowed, except CLOPE with $r=1$, in which only 30 clusters were found for the whole session file. That's the reason for a very fast speed of less than 1 minute per iteration for the whole dataset. The execution time of LargeItem is roughly 3-5 times as that of CLOPE, while LargeItem uses about 2 times the memory of CLOPE for the cluster feature data.

To have some impression on the effectiveness of CLOPE on noisy data, we run CLOPE on the November session data with $r=1.5$ and a maximum number of 1,000 clusters. The resulting clusters are ordered by the number of transactions they contain. Table 1 shows the largest cluster (C1000) with 20,089 transactions and other two high quality clusters found by CLOPE. These three clusters are quite good, but in many of the other clusters, pages from different paths are grouped together. Some of these may actually reveal some common visiting patterns, while others may due to noises inherited in the web log. However, our results of the LargeItem algorithm are not very satisfying.

Table 1. Some clusters of CLOPE on the log data ($r=1.5$).

C781: $N=554, W=6, S=1083$ <code>~/lazzaro/sa/book/simple/index.html, occ=426</code> <code>~/lazzaro/sa/index.html, occ=332</code> <code>~/lazzaro/sa, occ=170</code> <code>~/lazzaro/sa/book/index.html, occ=120</code> <code>~/lazzaro/sa/video/index.html, occ=26</code> <code>~/lazzaro/sa/sfman/user/network/index.html, occ=9</code>
C815: $N=619, W=6, S=1172$ <code>~/russell/aima.html, occ=388</code> <code>~/russell/code/doc/install.html, occ=231</code> <code>~/russell/code/doc/overview.html, occ=184</code> <code>~/russell/code/doc/user.html, occ=158</code> <code>~/russell/intro.html, occ=150</code> <code>~/russell/aima-bib.html, occ=61</code>
C1000: $N=20089, W=2, S=22243$ <code>/, occ=19517</code> <code>/Students/Classes, occ=2726</code>

* number after page name is the occurrence in the cluster

5. RELATED WORK

There are many works on clustering large databases, e.g. CLARANS [12], BIRCH [14], DBSCAN [4], CLIQUE [1]. Most of them are designed for low dimensional numerical data, exceptions are CLIQUE which finds dense subspaces in higher dimensions.

Recently, many works on clustering large categorical databases began to appear. The k-modes [10] approach represents a cluster of categorical value with the vector that has the minimal distance to all the points. The distance in k-modes is measured by number of common categorical attributes shared by two points, with optional weights among different attribute values. Han et.al. [8] use association rule hypergraph partitioning to cluster items in large transactional database. STIRR [6] and CACTUS [5] also model categorical clustering as a hypergraph-partitioning problem, but these approaches are more suitable for database made up of tuples. ROCK [7] uses the number of common neighbors between two transactions for similarity measure, but the computational cost is heavy, and sampling has to be used when scaling to large dataset.

³ Those non-directory requests having extensions other than “.s[html]”.

The most similar work to CLOPE is LargeItem [13]. However, our experiments show that CLOPE is able to find better clusters, even at a faster speed. Moreover, CLOPE requires only one parameter, repulsion, which gives the user much control over the approximate number of the resulting clusters, with little domain knowledge. The minimal support θ and the weight w of LargeItem are more difficult to determine. Our sensitivity tests of these two algorithms also show that CLOPE is less sensitive than LargeItem to the order of the input data.

Moreover, many works on document clustering are quite related with transactional data clustering. In document clustering, each document is represented as a weighted vector of words in it. Clustering is carried out also by optimizing a certain criterion function. However, document clustering tends to assume different weights on words with respect to their frequencies. See [15] for some common approaches in document clustering.

Also, there are some similarities between transactional data clustering and association analysis [2]. Both of these two popular data mining techniques can reveal some interesting properties of item co-occurrence and relationship in transactional databases. Moreover, current approaches [9] for association analysis needs only very few scans of the database. However, there are differences. On the one hand, clustering can give a general overview property of the data, while association analysis only finds the strongest item co-occurrence pattern. On the other hand, association rules are actionable directly, while clustering for large transactional data is not enough, and are mostly used as preprocessing phrase for other data mining tasks like association analysis.

6. CONCLUSION

In this paper, a novel algorithm for categorical data clustering called CLOPE is proposed based on the intuitive idea of increasing the height-to-width ratio of the cluster histogram. The idea is generalized with a repulsion parameter that controls tightness of transactions in a cluster, and thus the resulting number of clusters. The simple idea behind CLOPE makes it fast, scalable, and memory saving in clustering large, sparse transactional databases with high dimensions. Our experiments show that CLOPE is quite effective in finding interesting clusterings, even though it doesn't specify explicitly any inter-cluster dissimilarity metric. Moreover, CLOPE is not very sensitive to data order, and requires little domain knowledge in controlling the number of clusters. These features make CLOPE a good clustering as well as preprocessing algorithm in mining transactional data like market basket data and web usage data.

7. ACKNOWLEDGEMENTS

We are grateful to Rajeev Rastogi, Vipin Kumar for providing us the ROCK code and the technical report version of [7]. We wish to thank the providers of the UCI ML Repository and Web log files of <http://www.cs.berkeley.edu/>. We also wish to thank the authors of [13] for their help. Comments from the three anonymous referees are invaluable for us to prepare the final version.

8. REFERENCES

- [1] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. SIGMOD'98, Seattle, Washington, June 1998.
- [2] Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In Proc. SIGMOD'93, Washington, D.C., 1993.
- [3] Cooley, R., Mobasher, B., and Srivastava, J. Data preparation for mining world wide web browsing patterns. Knowledge and Information Systems, 1(1):5-32, 1999.
- [4] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proc. KDD'96, Portland, Oregon, 1996.
- [5] Ganti, V., Gehrke, J., and Ramakrishnan, R. CACTUS: Clustering categorical data using summaries. In Proc. KDD'99, San Diego, CA, 1999.
- [6] Gibson, D., Kleinberg, J., Raghavan, P. Clustering categorical data: an approach based on dynamical systems. In Proc. VLDB'98, New York, NY, 1998.
- [7] Guha, S., Rastogi, R., and Shim, K. ROCK: A robust clustering algorithm for categorical attributes. In Proc. ICDE'99, Sydney, Australia 1999.
- [8] Han, E.H., Karypis G., Kumar, V., and Mobashad, B. Clustering based on association rule hypergraphs. In Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997.
- [9] Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. In Proc. SIGMOD '00, Dallas, TX, 2000.
- [10] Huang Z. A fast clustering algorithm to cluster very large categorical data sets in data mining. In Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997.
- [11] MacQueen, J.B. Some methods for classification and analysis of multivariate observations. In Proc. 5th Berkeley Symposium on Math. Stat. and Prob., 1967.
- [12] Ng, R.T., and Han, J.W. Efficient and effective clustering methods for spatial data mining. In Proc. VLDB'94, Santiago, Chile, 1994.
- [13] Wang, K., Xu, C., and Liu, B. Clustering transactions using large items. In Proc. CIKM'99, Kansas, Missouri, 1999.
- [14] Zhang, T., Ramakrishnan, R., and Livny, M. BIRCH: An efficient data clustering method for very large databases. In Proc SIGMOD'96, Montreal, Canada, 1996.
- [15] Zhao, Y. and Karypis, G. Criterion functions for document clustering: experiments and analysis. Tech. Report #01-40, Department of Comp. Sci. & Eng., U. Minnesota, 2001. Available as: <http://www-users.itlabs.umn.edu/~karypis/publications/Papers/Postscript/vscluster.ps>