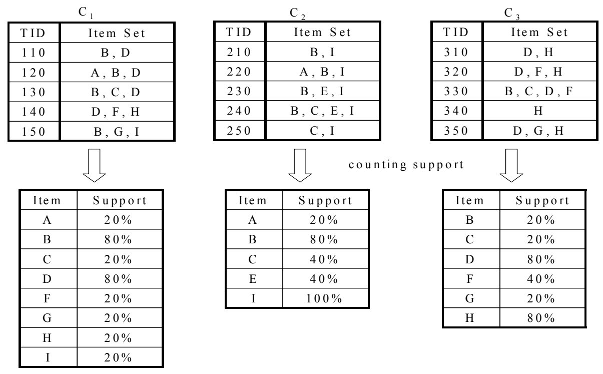
An Efficient Clustering Algorithm for Market Basket Data Based on Small Large Ratios

Ching-Huang Yun and Kun-Ta Chuang and Ming-Syan Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: chyun@arbor.ee.ntu.edu.tw, doug@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract

In this paper, we devise an efficient algorithm for clustering market-basket data items. In view of the nature of clustering market basket data, we devise in this paper a novel measurement, called the small-large (abbreviated as SL) ratio, and utilize this ratio to perform the clustering. With this SL ratio measurement, we develop an efficient clustering algorithm for data items to minimize the SL ratio in each group. The proposed algorithm not only incurs an execution time that is significantly smaller than that by prior work but also leads to the clustering results of very good quality.

Keywords —Data mining, clustering analysis, market-basket data, small-large ratios.



C_1

TID	Item Set
110	B, D
120	A, B, D
130	B, C, D
140	D, F, H
150	B, G, I

C_2

TID	Item Set
210	B, I
220	A, B, I
230	B, E, I
240	B, C, E, I
250	C, I

C_3

TID	Item Set
310	D, H
320	D, F, H
330	B, C, D, F
340	H
350	D, G, H

counting support

Item	Support
A	20%
B	80%
C	20%
D	80%
F	20%
G	20%
H	20%
I	20%

Item	Support
A	20%
B	80%
C	40%
E	40%
I	100%

Item	Support
B	20%
C	20%
D	80%
F	40%
G	20%
H	80%

Figure 1. An example database for clustering market-basket data.

1 Introduction

Mining of databases has attracted a growing amount of attention in database communities due to its wide applicability to improving marketing strategies [3][4]. Among others, data clustering is an important technique for exploratory data analysis [5][6]. In essence, clustering is meant to divide a set of data items into some proper groups in such a way that items in the same group are as similar to one another as possible. Market-basket data analysis has been well addressed in mining association rules for discovering the set of large items. Large items refer to frequently purchased items among all transactions and a transaction is represented by a set of items purchased [2]. Different from the traditional data, the features of market basket data are known to be high dimensionality, sparsity, and with massive outliers [7]. The authors in [8] proposed an algorithm for clustering market-basket data by utilizing the concept of large items to divide the transactions into clusters such that similar transactions are in the same cluster and dissimilar transactions are in different clusters. This algorithm in [8] will be referred to as algorithm Basic in this paper, and will be used for comparison purposes. An example database for clustering market-basket data is shown in Figure 1.

In view of the nature of clustering market basket data, we devise in this paper a novel measurement, called the small-large (abbreviated as SL) ratio, and utilize this ratio to perform the clustering. The *support* of an item i in a cluster C, $Sup_C(i)$, is defined as the percentage of transactions which contain this item i in cluster C. For the clustering U_0 shown in Figure 1, the support $Sup_{C_1}(A)$ is 20% and $Sup_{C_1}(B)$ is 80%. An item in a cluster is called a large item if the support of that item exceeds a pre-specified *minimum support S* (i.e., an item that appeared in a sufficient number of transactions). On the other hand, an item in a group is called a small item if the support of that item is less than a pre-specified *maximum ceiling E* (i.e., an item that appeared in a limited number of transactions). To model the relationship between minimum support S and maximum ceiling E, the damping factor λ is defined as the ratio of E to S, i.e.,

(Minimum Support S = 60%), (Maximal Ceiling E = 30%)			
Cluster	Large	Middle	Small
C_1	B, D		A, C, F, G, H, I
C_2	B, I	C, E	A
C_3	D, H	F	B, C, G
	Intra(U_0) = 7 Inter (U_0) = 2 Cost (U_0) = 9		

C_1

TID	Item Set	SL ratio
110	B, D	0/2
120	A, B, D	1/2
130	B, C, D	1/2
140	D, F, H	2/1
150	B, G, I	2/1

C_2

TID	Item Set	SL ratio
210	B, I	0/2
220	A, B, I	1/2
230	B, E, I	0/2
240	B, C, E, I	0/2
250	C, I	0/1

C_3

TID	Item Set	SL ratio
310	D, H	0/2
320	D, F, H	0/2
330	B, C, D, F	2/1
340	H	0/1
350	D, G, H	1/2

SLR Threshold $\alpha = 3/2$

Figure 2. The large, middle, and small items in clusters, and the corresponding SL ratios of transactions.

$\lambda = \frac{E}{S}$. In addition, an item is called a middle item if it is neither large nor small. For the supports of the items shown in Figure 1, if S = 60% and E = 30%, we can obtain the large, middle, and small items shown in Figure 2. In $C_2 = \{210, 220, 230, 240, 250\}$, B and I are large items. In addition, C and E are middle items and A is a small item.

Clearly, the portions of large and small items represent the quality of the clustering. Explicitly, the ratio of the number of small items to that of large items in a group is called small-large ratio of that group. Clearly, the smaller the SL ratio, the more similar the items in that group are. With this SL ratio measurement, we develop an efficient clustering algorithm, algorithm SLR (standing for Small-Large Ratio), for data items to minimize the SL ratio in each group. It is shown by our experimental results that by utilizing the SL ratio, the proposed algorithm is able to cluster the data items very efficiently.

This paper is organized as follows. Preliminaries are given in Section 2. In Section 3, an algorithm, referred to as *algorithm SLR (Small-Large Ratio)*, is devised for clustering market-basket data. Experimental studies are conducted in Section 4. This paper concludes with Section 5.

2 Preliminaries

We investigate the problem of clustering market-basket data, where the market-basket data is represented by a set of transactions. A database of transactions is denoted by $D = \{t_1, t_2, ..., t_h\}$, where each transaction t_i is a set of items $\{i_1, i_2, ..., i_h\}$. For the example shown in Figure 1, we are given a predetermined clustering $U_0 =< C_1, C_2, C_3 >$, where $C_1 = \{110, 120, 130, 140, 150\}$, $C_2 = \{210, 220, 230, 240, 250\}$, and $C_3 = \{310, 320, 330, 340, 350\}$.

2.1 Large Items and Small Items

The concept of large items is first introduced in mining association rules [2]. In [8], using large items as similarity measure of a cluster is utilized in clustering transactions. Specifically, large items in cluster C_j are the items frequently purchased by the customers in cluster C_j. In other words, large items are popular items in a cluster and thus contribute to similarity in a cluster. While rendering the clustering of fine quality, it is noted that the execution efficiency of the algorithm in [8] could be further improved due to its relatively inefficient steps in the refinement phase. This could be partly due to the reason that the similarity measurement used in [8] does not take into consideration the existence of small items. To remedy this, a maximal ceiling E is proposed in this paper for identifying the items of rare occurrences. If an item whose support is below a specified maximal ceiling E, that item is called a small item. Hence, small items in a cluster contribute to dissimilarity in a cluster. In this paper, the similarity measurement of transactions is derived from the ratio of the number of large items to that of small items. In the example shown in Figure 1, with the minimum support S = 60% and the maximum ceiling E = 30%, we can obtain the large, middle, and small items by counting their supports. In C_1, item B is large because its support value is 80% (appearing in TID 110, 120,130, and 150) which exceeds the minimum support S. However, item A is small in C_1 because its support is 20% which is less than the maximum ceiling E.

2.2 Cost Function

We use $La_I(C_j, S)$ to denote the set of large items with respect to attribute I in C_j, and $Sm_I(C_j, E)$ to denote the set of small items with respect to attribute I in C_j. For a clustering $U = \{C_1, ..., C_k\}$, the corresponding cost for attribute I has two components: the intra-cluster cost $Intra_I(U)$ and the inter-cluster cost $Inter_I(U)$, which are described in detail below.

Intra-Cluster Cost: The intra-cluster item cost is meant to represent for intra-cluster item-dissimilarity and is measured by the total number of small items, where a small item is an item whose support is less than the maximal ceiling E. Explicitly, we have

$$Intra_I(U) = | \cup_{j=1}^{k} Sm_I(C_j, E)|.$$

Note that we did not use $\Sigma_{j=1}^{k} |Sm_I(C_j, E)|$ as the intra-cluster cost since the use of $\Sigma_{j=1}^{k} |Sm_I(C_j, E)|$ may cause the algorithm to tend to put all records into a single or few clusters even though they are not similar. For example, suppose that there are two clusters that are not similar but share some small items. If large items remain large after the merging, merging these two clusters will reduce $\Sigma_{j=1}^{k}|Sm_I(C_j, E)|$ because each small item previously counted twice is now counted only once. However, this merging is incorrect because sharing of small items should not be considered as similarity. For the clustering U_0 shown in Figure 2, the small items of C_1 are {A, C, F, G, H, I}. In addition, the small item of C_2 is {A} and the small items of C_3 are {B, C, G}. Thus, the intra-cluster cost $Intra_I(U_0)$ is 7.

Inter-Cluster Cost: The inter-cluster item cost is to represent inter-cluster item-similarity and is measured by the duplication of large items in different clusters, where a large item is an item whose support exceeds the minimum support S. Explicitly, we have

$$Inter_I(U) = \Sigma_{j=1}^{k}|La_I(C_j, S)| - |\cup_{j=1}^{k} La_I(C_j, S)|.$$

Note that this measurement will inhibit the generation of similar clusters. For the clustering U_0 shown in Figure 2, the large items of C_1 are {B, D}. In addition, the large items of C_2 are {B, I} and the large items of C_3 are {D, H}. As a result, $\Sigma_{j=1}^{k}|La_I(C_j, S)|$ = 6 and $|\cup_{j=1}^{k} La_I(C_j, S)|$ = 4. Hence, the inter-cluster cost $Inter_I(U_0)$ = 2.

Total Cost: Both the intra-cluster item-dissimilarity cost and the inter-cluster item-similarity cost should be considered as the total cost incurred. Without loss of generality, a weight w is specified for the relative importance of these two terms. The definition of item cost $Cost_I(U_0)$ with respect to attribute I is:

$$Cost_I(U_0) = w * Intra_I(U_0) + Inter_I(U_0).$$

If the weight $w > 1$, $Intra_I(U_0)$ is more important than $Inter_I(U_0)$, and vice versa. In our model, we let $w = 1$. Thus, for the clustering U_0 shown in Figure 2, the $Cost_I(U_0)$ is $7 + 2 = 9$.

2.3 Objective of Clustering Market-Basket Data

The objective of clustering market-basket data is *"We are given a database of transactions, a minimum support, and a maximum ceiling. Then, we would like to determine a clustering U such that the total cost is minimized*". The procedure of clustering algorithm we shall present includes two phases, namely, the allocation phase and the refinement phase. In the allocation phase, the database is scanned once and each transaction is allocated to a cluster based on the purpose of minimizing the cost. The method of allocation phase is straightforward and the approach taken in [8] will suffice. In the refinement phase, each transaction will be evaluated for its status to minimize the total cost. Explicitly, a transaction is moved from one cluster to another cluster if that movement will reduce the total cost of clustering. The refinement phase repeats until no further movement is required. The goal of this paper focuses on designing an efficient algorithm for the refinement phase.

3 Algorithm SLR for Clustering Market-Basket Data

In this section, we devise algorithm SLR (Small-Large Ratio) which essentially utilizes the measurement of the *small-large ratio (SL ratio)* for clustering market-basket data. For a transaction t with one attribute I, $|L_I(t)|$ represents the number of the large items in t and $|S_I(t)|$ represents the number of the small items in t. The SL ratio of t with attribute I in cluster C_i is defined as:

$$SLR_I(C_i, t) = \frac{|S_I(t)|}{|L_I(t)|}.$$

For the clustering shown in Figure 1, $C_1 = \{110, 120, 130, 140, 150\}$, $C_2 = \{210, 220, 230, 240, 250\}$, and $C_3 = \{310, 320, 330, 340, 350\}$. Figure 2 shows that the minimum support S = 60% and the maximal ceiling E = 30%. For TID 120, we have two large items {B, D} and one small item {A}. Thus, the SL ratio of TID 120 is $SLR_{Item}(C_1, 120) = \frac{1}{2} = 0.5$. Similarly, the SL ratio of TID 240 is $SLR_{Item}(C_2, 240) = \frac{2}{2} = 1$, because TID 240 has 2 large items {B, I} and 2 small items {C, E}. As mentioned before, although algorithm Basic utilizes the large items for similarity measurement, algorithm Basic is exhaustive in the decision procedure of moving a transaction t to cluster C_j in current clustering $U = \{C_1, ..., C_k\}$. For each transaction t, algorithm Basic must compute all costs of new clusterings when t is put into another cluster. In contrast, by utilizing the concept of small-large ratios, algorithm SLR can efficiently determine the next cluster for each transaction in an iteration, where an iteration is a refinement procedure from one clustering to the next clustering.

3.1 Description of Algorithm SLR

Figure 3 shows the main program of algorithm SLR, which includes two phases: the allocation phase and the refinement phase. Similarly to algorithm Basic [8], in the allocation phase, each transaction t is read in sequence. Each transaction t can be assigned to an existing cluster or a new cluster will be created to accommodate t for minimizing the total cost of clustering. For each transaction, the initially allocated cluster identifier is written back to the file. However, different from algorithm Basic, algorithm SLR compares the SL ratios with the pre-specified *SLR threshold* α to determine the best cluster for each transaction. Note that some transactions might not be suitable in the current clusters. Hence, we define an *excess transaction* as a transaction whose SL ratio exceeds the SLR threshold α. In each iteration of the refinement phase, algorithm SLR first computes the support values of items for identifying the large items and the small items in each cluster. Then, algorithm SLR searches every cluster to move excess transactions the *excess pool*, where all excess transactions are collected together. After collecting all excess transactions, we compute the intermediate support values of items for identifying the large items and the small items in each cluster again. Furthermore, empty clusters are removed. In addition, we read each transaction t_p from the excess pool. In line 8 to line 14 of the refinement phase shown in Figure 3, we shall find for each transaction the best cluster that is the cluster where that transaction has the minimal SL ratio after all clusters are considered. If that ratio is smaller than the SLR threshold, we then move that transaction from the excess pool to the best cluster found. However, if there is no appropriate cluster found for that transaction t_p, t_p will remain in the excess pool. If there is no movement in an iteration after all transactions are scanned in the excess pool, the refinement phase terminates. Otherwise, the iteration continues until there is no further movement identified. After the refinement phase completes, there could be some transactions still in the excess pool that are not thrown into any appropriate cluster. These transactions will be deemed outliers in the final clustering result. In addition, it is worth mentioning that algorithm SLR is able to support the incremental clustering in such a way that those transactions added dynamically can be viewed as new members in the excess pool. Then, algorithm SLR will allocate them into the appropriate clusters based on their SL ratios in existing clusters. By treating the incremental transactions as new members in the excess pool, algorithm SLR can be applied to clustering the incremental data efficiently.

3.2 Illustrative Example of SLR

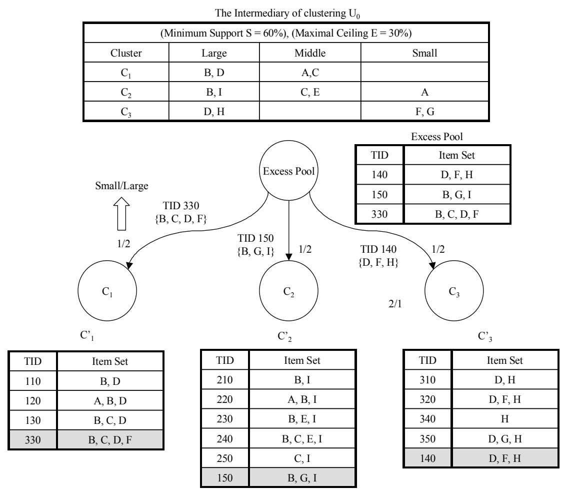
Suppose the clustering $U_0 =< C_1, C_2, C_3 >$ shown in Figure 1 is the clustering resulted by the allocation phase. The cost of U_0 examined by the similarity measurement is shown Figure 2. In this experiment, the minimum support S = 60%, the maximal ceiling E = 30%, and the SLR threshold $\alpha = \frac{3}{2}$. In the refinement phase shown in Figure 4, algorithm SLR computes the SL ratio for each transaction and reclusters the transactions whose SL ratios exceed α. Figure 5 is the final clustering $U_1 =< C_1', C_2', C_3' >$ obtained by applying algorithm SLR to the clustering U_0. First, algorithm SLR scans the database and counts the supports of items shown in Figure 1. In C_1, the support of item A is 20% and the support of item B is 80%. Then, algorithm SLR identifies the large and small items shown

```
/*Allocation phase*/
1) While not end of the file do {
2)    Read the next transaction t;
3)    Allocate t to an existing or a new cluster $C_i$ to minimize Cost(U);
4)    Write <t,$C_i$>;
5) } /*while*/

/*Refinement phase*/
1) do {
2)    not_moved=false;
3)    calculate each cluster's minimum support, large items and small items;
4)    move out all excess transactions from each cluster to excess pool;
5)    eliminate any empty cluster;
6)    afresh calculate each cluster's minimum support, large items and small items;
7)    while not end of excess pool {
8)       Read the next transaction $t_p$ from excess pool;
9)       Search for the best choosing cluster $C_j$ that $t_p$ will have the smallest SLR
            in $C_j$;
10)      if find $C_j$ {
11)         remove $t_p$ from excess pool;
12)         move $t_p$ to cluster $C_j$;
13)         not_moved=true;
14)      } /*if*/
15)   } /*while*/
16) } while (not_moved); /*do*/
```

Figure 3. The overview of algorithm SLR.

in Figure 2. In C_1, item A is a small item and item B is a large item. For the transactions in each cluster, algorithm SLR computes their SL ratios in that cluster. In C_1, the large items are {B, D} and the small items are {A, C, F, G, H, I}. For transaction TID 120, item {A} is a small item and items {B, D} are large items. Thus, the SL ratio of TID 120 is $SLR_{Item}(C_1, 120) = \frac{1}{2}$ which is smaller than α. However, for transaction TID 140, items {F, H} are small items and item {D} is the only large one. The SL ratio of TID 140 is $SLR_{Item}(C_1, 140) = \frac{2}{1}$, larger than α. After the SL ratios of all transactions are determined, algorithm SLR shall identify the excess transactions and move them into the excess pool. Three transactions, i.e., TIDs 140, 150, and 330, are identified as the excess transactions as shown in Figure 2. After collecting all excess transactions, we compute the intermediate support values of items for identifying the large items and the small items in each cluster again. The intermediate clustering of U_0 is shown in Figure 4. For each transaction in the excess pool, algorithm SLR will compute its SL ratios associated with all clusters, except the cluster that transaction comes from. Note that an item that is not shown in the cluster C_i can be viewed as a small item because its support will be one when the corresponding transaction is added into C_i. For transaction TID 140 moved from C_1, $SLR_{Item}(C_2, 140) = \frac{3}{0} = \infty$ with three small items {D, F, H} in C_2. On the other hand, $SLR_{Item}(C_3, 140) = \frac{1}{2}$ with one small item {F} and two large items {D, H} in C_3. For transaction TID 140,



The Intermediary of clustering U_0

(Minimum Support S = 60%), (Maximal Ceiling E = 30%)			
Cluster	Large	Middle	Small
C_1	B, D	A,C	
C_2	B, I	C, E	A
C_3	D, H		F, G

Excess Pool

TID	Item Set
140	D, F, H
150	B, G, I
330	B, C, D, F

C'1

TID	Item Set
110	B, D
120	A, B, D
130	B, C, D
330	B, C, D, F

C'2

TID	Item Set
210	B, I
220	A, B, I
230	B, E, I
240	B, C, E, I
250	C, I
150	B, G, I

C'3

TID	Item Set
310	D, H
320	D, F, H
340	H
350	D, G, H
140	D, F, H

Figure 4. Using small-large ratio to recluster the transactions by algorithm SLR.

C'_1

Item	Support
A	25%
B	100%
C	50%
D	100%
F	25%

C'_2

Item	Support
A	16.7%
B	83.3%
C	33.3%
E	33.3%
G	16.7%
I	100%

C'_3

Item	Support
D	80%
F	40%
G	20%
H	100%

(Minimum Support S = 60%), (Maximal Ceiling E = 30%)			
Cluster	Large	Middle	Small
C'_1	B, D	C	A, F
C'_2	B, I	C, E	A, G
C'_3	D, H	F	G
	Intra(U_1) = 3 Inter (U_1) = 2 Cost (U_1) = 5		

Figure 5. The clustering $U_1 =< C'_1, C'_2, C'_3 >$ obtained by algorithm SLR.

the smallest SL ratio is $SLR_{Item}(C_3, 140) = \frac{1}{2}$ which is smaller than $\alpha = \frac{3}{2}$. Thus, transaction TID 140 is reclustered to C_3. Figure 4 shows that algorithm SLR utilizes the SL ratios to recluster transactions to the most appropriate clusters. The resulting clustering is $U_1 =< C'_1, C'_2, C'_3 >$. In the new clustering, algorithm SLR will compute the support values of items for all clusters. Figure 5 shows the supports of the items in C'_1, C'_2,and C'_3. Algorithm SLR proceeds until no more transaction is reclustered. The clustering U_1 is also the final clustering for this example and the final cost $Cost_I(U_1) = 5$, which is smaller than the initial cost $Cost_I(U_0) = 9$.

4 Experimental Results

To assess the performance of algorithm SLR and algorithm Basic, we conducted several experiments for clustering various data. We comparatively analyze the quality and performance between algorithm SLR and algorithm Basic in the refinement phase.

4.1 Data Generation

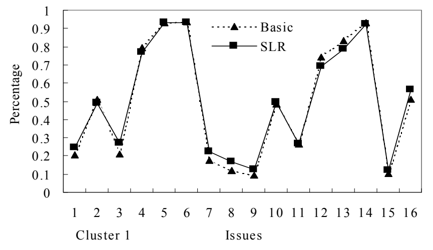
We take the real data sets of the United States Congressional Votes records in 1984 [1] for performance evaluation. The file of 1984 United States congressional votes contains 435 records, each of which includes 16 binary attributes corresponding to every congressman's vote on 16 key issues, e.g., the problem of the immigration, the duty of export, and the educational spending, and so on. There are 168 records for Republicans and 267 for Democrats. We set the minimum support to 60%, which is the same as the minimum support setting in [8] for comparison purposes.

To provide more insight into this study, we use a well-known market-basket synthetic data in [2], as the synthetic data for performance evaluation. This code will generate volumes of transaction data over a large range of data characteristics. These transactions mimic the transactions in the real world retailing environment. The size of the transaction is picked from a Poisson distribution with mean |T|, which is set to 5 in our Experiments. In addition, the average size of the maximal potentially large item sets, denoted by |I|, is set to 2. The number of maximal potential large item sets, denoted by |L|, is set to 2000. The number of items, denoted by |N|, is set to 1000 as default.

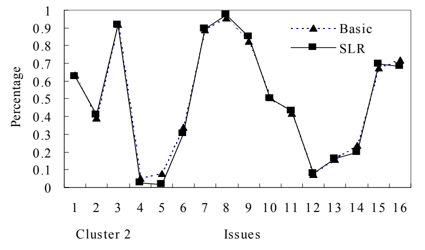
4.2 Performance Study

In the experiment for the real data, $S = 0.6$ and $\alpha = 2.5$, and λ varies from 0.4 to 1, where λ is the damping factor. Figure 6 shows the results of two clusters, cluster 1 for Republicans and cluster 2 for Democrats. It shows that these two results are similar to each other in the percentages of the issues in cluster 1 and cluster 2. Recall that an iteration is a refinement procedure from one clustering to the next clustering. Figure 7 shows the comparison of the execution time between algorithm SLR and algorithm Basic in each iteration. It can be seen that although algorithm SLR has one more iteration than algorithm Basic, the execution time of algorithm SLR is much shorter than that of algorithm Basic in every iteration.

We next use the synthetic data mentioned above in the following experiments. It is shown by Figure 8 that as the database size increases, the execution time of algorithm Basic increases rapidly whereas that of algorithm SLR in-



(a) For Republicans



(b) For Democrats

Figure 6. The percentage of the issues in cluster 1 and cluster 2.

creases linearly, indicating the good scale-up feature of algorithm SLR.

5 Conclusion

In view of the nature of clustering for market basket data, we devised in this paper a novel measurement, called the small-large ratio. We have developed an efficient clustering algorithm for data items to minimize the SL ratio in each group. The proposed algorithm is able to cluster the data items very efficiently. This algorithm not only incurs an execution time that is significantly smaller than that by prior work but also leads to the clustering results of very good quality.

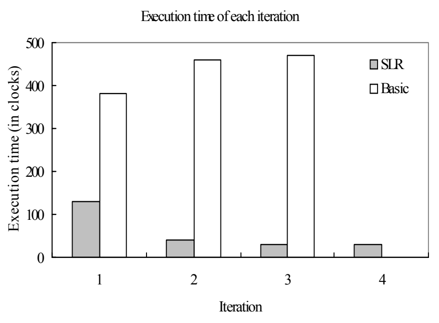


Figure 7. Execution time of algorithm SLR and algorithm Basic in each iteration.

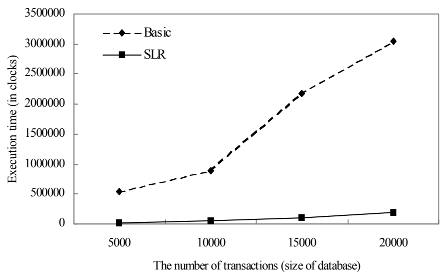


Figure 8. Execution time of algorithm SLR and algorithm Basic as the number of transactions $|D|$ varies.

Acknowledgments

The authors were supported in part by the Ministry of Education Project No. 89-E-FA06-2-4-7 and the National Science Council, Project No. NSC 89-2218-E-002-028 and NSC 89-2219-E-002-028, Taiwan, Republic of China.

References

[1] UCI Machine Learning Repository. *http:://www.ics.uci.edu/~mlearn/MLRepository.html*.

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

[3] A. G. Buchner and M. Mulvenna. Discovery Internet Marketing Intelligence through Online Analytical Web Usage Mining. *ACM SIGMOD Record*, 27(4):54–61, Dec. 1998.

[4] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from a Database Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–833, 1996.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: a Review. *ACM Computing Surveys*, 31(3):264–323, Sep. 1999.

[6] D. A. Keim and A. Hinneburg. Clustering Techniques for the Large Data Sets - ¿From the Past to the Future. *Tutorial notes for ACM SIGKDD 1999 international conference on Knowledge discovery and data mining*, pages 141–181, Aug. 1999.

[7] A. Strehl and J. Ghosh. A Scalable Approach to Balanced, High-dimensional Clustering of Market-baskets. *Proceedings of the 7th International Conference on High Performance Computing*, December 2000.

[8] K. Wang, C. Xu, and B. Liu. Clustering Transactions Using Large Items. *ACM CIKM International Conference on Information and Knowledge Management*, pages 483–490, Nov. 1999.