

Self-Tuning Clustering: An Adaptive Clustering Method for Transaction Data

Ching-Huang Yun¹, Kun-Ta Chuang², and Ming-Syan Chen¹

¹ Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC

mschen@cc.ee.ntu.edu.tw, chyun@arbor.ee.ntu.edu.tw

² Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan, ROC
doug@arbor.ee.ntu.edu.tw

Abstract. In this paper, we devise an efficient algorithm for clustering market-basket data items. Market-basket data analysis has been well addressed in mining association rules for discovering the set of large items which are the frequently purchased items among all transactions. 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. In view of the nature of clustering market basket data, we present a measurement, called the small-large (SL) ratio, which is in essence the ratio of the number of small items to that of large items. Clearly, the smaller the SL ratio of a cluster, the more similar to one another the items in the cluster are. Then, by utilizing a self-tuning technique for adaptively tuning the input and output SL ratio thresholds, we develop an efficient clustering algorithm, *algorithm STC* (standing for *Self-Tuning Clustering*), for clustering market-basket data. The objective of algorithm STC is “Given a database of transactions, determine a clustering such that the average SL ratio is minimized.” We conduct several experiments on the real data and the synthetic workload for performance studies. It is shown by our experimental results that by utilizing the self-tuning technique to adaptively minimize the input and output SL ratio thresholds, algorithm STC performs very well. Specifically, algorithm STC not only incurs an execution time that is significantly smaller than that by prior works but also leads to the clustering results of very good quality.

Keywords: Data mining, clustering market-basket data, small-large ratios, adaptive self-tuning.

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].

Among others, data clustering is an important technique for exploratory data analysis [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. Most clustering techniques utilize a pairwise similarity for measuring the distance of two data points. Recently, there has been a growing emphasis on clustering very large datasets to discover useful patterns and correlations among attributes [4] [10]. Note that clustering is a very service dependent issue and its potential applications call for their own specific requirements.

Itemset data, referred to as market-basket data, has been well studied in mining association rules for discovering the set of frequently purchased items [2]. Different from the traditional data, the features of market basket data are known to be high dimensionality, sparsity, and with massive outliers. It is noted that there are several clustering technologies which addressed the issue of clustering market-basket data [5][7][8][9]. ROCK [5] is an agglomerative hierarchical clustering algorithm by treating market-basket data as categorical data and using the links between the data points for clustering categorical data. The authors in [7] proposed an algorithm by using large items as the similarity measurement to divide the transactions into clusters such that similar transactions are in the same clusters. OAK [8] combines hierarchical and partitional clustering techniques. The work in [9] utilized a predetermined ratio for clustering market-basket data.

In view of the nature of clustering market-basket data, we present in [9] a measurement, called the *small-large (SL) ratio*, which is in essence the ratio of the number of small items to that of large items. With their formal definitions given in Section 2, a *large item* is basically an item with frequent occurrence in transactions whereas a *small item* is an item with infrequent occurrence in transactions. Clearly, the smaller the SL ratio of a cluster, the more similar to one another the items in the cluster are. Then, by utilizing a self-tuning technique for adaptively tuning the input and output SL ratio thresholds, we develop an efficient clustering algorithm, referred to as *algorithm STC* (standing for *Self-Tuning Clustering*), for clustering market-basket data. Algorithm STC consists of three phases, namely, the *pre-determination phase*, the *allocation phase*, and the *refinement phase*. In the pre-determination phase, the *minimum support* S and the *maximum ceiling* E are calculated according to a given parameter, called *SL distribution rate* β . In the allocation phase, algorithm STC uses the minimum support S to identify the large items and uses the maximum ceiling E to identify the small items. Explicitly, algorithm STC scans the database and allocates each transaction to a cluster for minimizing the SL ratio. In the refinement phase, each transaction will be evaluated for its status to minimize its SL ratio in the corresponding cluster. Algorithm STC utilizes two kinds of SL ratio thresholds, *output SL ratio threshold* α^o and *input SL ratio threshold* α^i , to evaluate the quality of the clustering. Explicitly, a transaction is moved from one cluster to the excess pool if its SL ratio is larger than α^o , and a transaction is moved from the excess pool to one cluster if the SL ratio is smaller than α^i . Detailed operations of STC will be described in Section 3 later. It is important to note that by utilizing $\alpha(U)$ to tune both α^i and α^o , STC is able to minimize the SL ratios of

transactions in individual clusters, thereby improving the quality of clustering. As will be shown by our experimental results, algorithm STC devised in this paper significantly outperforms previous efforts [5][7] in the execution efficiency and also the clustering quality for both synthetic and real market-basket data.

This paper is organized as follows. Preliminaries are given in Section 2. In Section 3, algorithm STC is devised for clustering market-basket data. Experimental studies are conducted in Section 4. This paper concludes with Section 5.

2 Problem Description

In this paper, the market-basket data is represented by a set of transactions. A database of transactions is denoted by $D = \{t_1, t_2, \dots, t_h\}$, where each transaction t_i is a set of items $\{i_1, i_2, \dots, i_h\}$. Similarly to the definition taken by [7], a clustering $U = \langle C_1, C_2, \dots, C_k \rangle$ is a partition of transactions, where C_i is a cluster consisting of a set of transactions. Note that the minimum support S and the maximum ceiling E are determined according to the SL distribution rate β in the pre-determination phase. The details of the pre-determination phase will be shown later.

2.1 Large Items and Small Items

The support of an item i in a cluster C , called $Sup_C(i)$, is defined as the percentage of transactions which contain this item i in cluster C . An item i in a cluster C is called a *large item* if $Sup_C(i)$ is larger than the minimum support S . On the other hand, an item j in a cluster C is called a *small item* if $Sup_C(j)$ is less than the maximum ceiling E . In addition, an item is called a middle item if it is neither large nor small.

2.2 Small-Large (SL) Ratio

Note that there are three kinds of SL ratios, i.e., *SL ratio of a transaction*, *SL ratio of a clustering*, and *average SL ratio*, to be calculated in the data clustering procedure.

SL Ratio of a Transaction: For a transaction t in cluster C_i , the SL ratio of t in cluster C_i is defined as:

$$R_{SL}(C_i, t) = \frac{|S(C_i, t)|}{|L(C_i, t)|},$$

where $|L(C_i, t)|$ represents the number of the large items in t and $|S(C_i, t)|$ represents the number of the small items in t .

SL Ratio of a Clustering: For a clustering $U = \langle C_1, \dots, C_p \rangle$, the SL ratio of U is defined as:

$$R_{SL}(U) = \sum_{i=1}^p \sum_{j=1}^{N^T(C_i)} R_{SL}(C_i, t_j),$$

where $N^T(C_i)$ is the number of transactions and t_j is the j th transaction in cluster C_i .

Average SL Ratio: For a clustering $U = \langle C_1, \dots, C_p \rangle$, the average SL ratio is defined as:

$$\alpha(U) = \frac{R_{SL}(U)}{N^T(U)},$$

where $N^T(U)$ is the number of transactions in clustering U .

2.3 Objective of Clustering Market-Basket Data

The objective of clustering market-basket data is “*Given a database of transactions, determine a clustering U such that the average SL ratio $\alpha(U)$ is minimized.*” For each cluster, the large items correspond to the products which are sold frequently. Such an information is usually useful for vendors to deal with market segmentation. Note that the clustering technique we devise aims to maximizing both the *intra-cluster similarity* and the *inter-cluster dissimilarity*.

Intra-Cluster Similarity: An item i is large if there are sufficient transactions containing i . These transactions are said to be similar to one another if they contain many common large items. On the other hand, an item j is small if there are relatively few transactions containing j . Thus, having fewer small items implies that these transactions are more similar to one another. To achieve the intra-cluster similarity of transactions, the number of large items is to be maximized whereas that of small items should be minimized in each cluster. The average SL ratio can hence be minimized.

Inter-Cluster Dissimilarity: To achieve the inter-cluster dissimilarity of transactions, a large item in one cluster is expected to be a small item in another cluster. Explicitly, if an item i is large in cluster C_a , there should be relatively few transactions in another cluster C_b containing i . By maximizing the number of large items and minimizing the number of small items, dissimilar transactions will be allocated to the different clusters. By increasing the inter-cluster dissimilarity, the average SL ratio is reduced.

3 Algorithm STC for Adaptively Clustering Market-Basket Data

By utilizing the concept of SL ratios, we devise an algorithms, called algorithm STC, for clustering market-basket data with the self-tuning technique to adaptively tune both the input and output SL ratio thresholds for iteratively minimizing $\alpha(U)$. Algorithm STC has three phases: the pre-determination phase, the allocation phase, and the refinement phase. An algorithmic form of algorithm STC can be found in the Appendix A for interested readers. In the pre-determination phase, the minimum support S and the maximum ceiling E are obtained by calculating the supports of items and identifying the related items according to SL distribution rate β . The procedure of algorithm STC is shown in Figure 1. First, STC counts the supports of items by scanning the database and the items are sorted according to their supports. Let $Count(\beta)$ be $N^I \times \beta$, where N^I is the number of total items. Then, in the sorted items, the support of the item whose support is the $Count(\beta)$ th largest one among all item supports is the minimum support S , and the support of the item whose support is the $Count(\beta)$ th smallest one is the maximum ceiling E .

After obtaining the minimum support S and the maximum ceiling E in the pre-determination phase, STC determines the initial clustering U_0 for transactions in the allocation phase. Then, each transaction is read in sequence and assigned to an existing cluster or a new cluster created to accommodate that transaction with the objective of minimizing the SL ratio. Explicitly, STC reads each transaction t from database D . In this paper, N_C is defined as the number of clusters and μ is defined as the upper bound of the number of clusters. In the case of $N_C < \mu$, t is assigned to an existing cluster C_i if $R_{SL}(C_i, t)$ is the smallest and $R_{SL}(C_i, t)$ is smaller than one. Otherwise, t is assigned to a new cluster and N_C is increased by one. If $N_C = \mu$, t is assigned to an existing cluster C_i so that $R_{SL}(C_i, t)$ is the smallest. After all transactions are allocated to the clusters, STC obtains the initial clustering $U_0 = \langle C_1, C_2, \dots, C_{N_C} \rangle$.

The self-tuning technique devised in this paper is exhibited in the refinement phase. After obtaining the initial clustering U_0 , STC adaptively tunes the input and output SL ratio thresholds for clustering market-basket data in the refinement phase. In the beginning of the refinement phase, algorithm STC will identify the large, middle, and small items in each cluster in accordance with their supports. In iteration 1, STC sets $\alpha_1^o = \alpha_1^i = \alpha(U_0)$. Then, STC moves the transactions, whose SL ratios are larger than α_1^o , from their clusters to the excess pool. The transactions in the excess pool are also called *excess transactions*. After all transactions with their SL ratios larger than α_1^o are moved from their clusters to the excess pool, STC obtains an intermediate clustering U_1' . For the intermediate clustering U_1' , STC counts the supports of items again to identify the large, middle, and small items in each cluster. Then, for each excess transaction t , algorithm STC calculates its SL ratio in each cluster based on the large and small items in that cluster of the intermediate clustering U_1' . If the smallest SL ratio of t is in cluster C_j and is smaller than α_1^i , STC shall move t from the excess pool to cluster C_j .

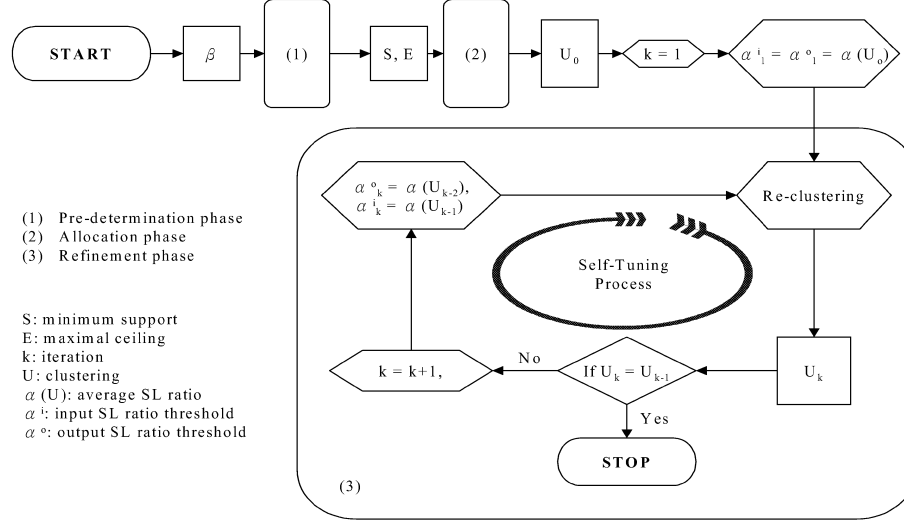


Fig. 1. The procedure of algorithm STC with the self-tuning process in the refinement phase.

After all excess transactions are evaluated in iteration 1, STC finishes the re-clustering and obtains the clustering U_1 . In each subsequent iteration, STC starts with the clustering found in the previous iteration, adaptively tunes input and output SL ratio thresholds, and then re-clusters the transactions in the clustering and the excess pool. Explicitly, in iteration k , algorithm STC sets output SL ratio threshold α_k^o to be $\alpha(U_{k-2})$ and input SL ratio threshold α_k^i to be $\alpha(U_{k-1})$. Recall that α_k^o is the threshold for moving the transactions from their clusters to the excess pool. If the SL ratio of a transaction t in the clustering U_{k-1} is larger than α_k^o , t will be moved to the excess pool. On the other hand, if the SL ratio of a transaction t in the excess pool is smaller than α_k^i , t will be moved from the excess pool and become a member in the clustering U_k . The procedure continues until no further movement is required (i.e., the average SL ratio $\alpha(U)$ cannot be reduced any more). As will be validated by the experimental studies later, with the advantage of adaptive tuning, algorithm STC is able to efficiently attain the clustering results of very high quality.

4 Experimental Results

To assess the comparison among algorithms, we conducted several experiments. The details of data generation are described in Section 4.1. In Section 4.2, the quality and performance comparisons among algorithms are discussed. Without adaptively tuning the input and output SL ratio thresholds, algorithm SLR will cluster market-basket data with the fixed output SL ratio threshold α^o and fixed

input SL ratio threshold α^i in the refinement phase. Explicitly, in iteration k , algorithm SLR sets $\alpha_k^o = \alpha_k^i = \alpha$, where α is called *fixed SL ratio threshold* and is given *a priori*. Rather than being determined by the SL distribution rate β as in STC, the minimum support S and the maximal ceiling E in SLR are also given *a priori*.

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. To provide more insight into this study, we use a well-known market-basket synthetic data generated by the IBM Quest Synthetic Data Generation Code [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. In addition, we take the real market-basket data from a large bookstore company for performance evaluation. In this real data set, there are $|D| = 98934$ transactions and $N^I = 58909$ items.

4.2 Performance Study

For the voting data, we conduct three experiments, from Experiment One to Experiment Three, for evaluating the clustering quality of algorithms. In algorithm STC, the SL distribution rate β is set to 30% and μ is set to 2. This algorithm in [7] will be referred to as algorithm Basic in our experimental studies. In algorithm Basic, the minimum support S is set to 60%, which is the same as the minimum support setting in [7] for comparison purposes. In algorithm ROCK, the threshold θ is set to 0.73, which is the same as the value setting in [5]. In algorithm SLR, we set the fixed SL ratio threshold $\alpha = 2.5$, $S = 60\%$, and $E = 60\%$. For the synthetic market-basket data, we conduct Experiment Four for evaluating the clustering performance of algorithms. For the experiment on synthetic data, we set $\beta = 30\%$ in algorithm STC, $S = 0.1\%$ in algorithm Basic, and $\theta = 0.73$ in algorithm ROCK. In addition, we set $\alpha = 2.5$, $S = 0.05$, and $E = 0.05$ in SLR. For the real market-basket data, we also conduct Experiments Five and Six to assess the performance of algorithms proposed.

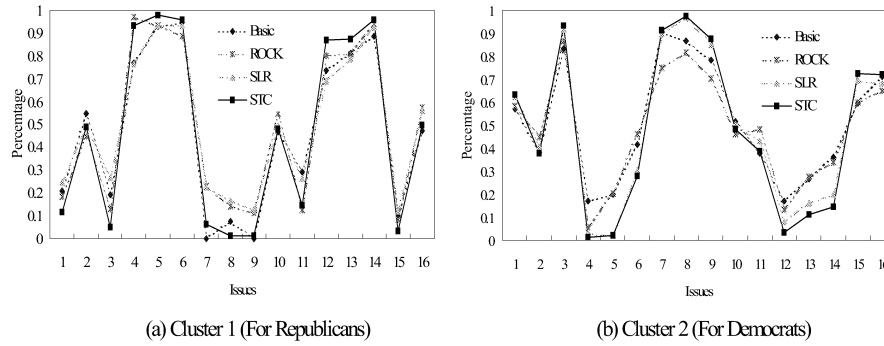
Experiment One: Evaluating clustering quality for congressional voting data. In Table 1, the quality of the clustering algorithms is evaluated by *noise ratios* of the clustering [5], where the *noise ratio* can be obtained by dividing the number of obscure politics in the clustering by the total number in the clustering. For example, the democrats is the obscure politics in cluster 1 and therefore the noise ratio of algorithm Basic is $\frac{46}{171} = 26.9\%$. The smaller the

Table 1. Comparison on the clustering quality for congressional voting data.

Cluster 1	No. of Republicans	No. of Democrats	Total No.	Noise Ratio
Basic	125	46	171	0.269
ROCK	144	22	166	0.133
SLR	161	62	223	0.278
STC	132	13	145	0.090
Cluster 2	No. of Republicans	No. of Democrats	Total No.	Noise Ratio
Basic	43	221	264	0.163
ROCK	5	201	206	0.024
SLR	6	205	211	0.028
STC	3	187	190	0.016

noise ratio, the better quality this algorithm has. With self-tuning techniques, it shows that STC emerges as the one with the best clustering quality.

Experiment Two: Evaluating the clustering quality of issues for congressional voting data. For each issue in the Congressional Votes records, each congressman can vote for the issue (marked as 1) or vote against the issue (marked as 0). Each record represents a transaction. For each issue, voting for the issue (or voting against the issue) represents an item in the clusters. In our opinion, for Congressional Votes records, the better clustering quality also means that the issues in each cluster should be closer to 1 (which means that the members in the cluster vote for the issue) or closer to 0 (which means that the members in the cluster vote against the issue). By minimizing the SL ratios of records in cluster 1 and cluster 2, it is shown in Figure 2 that STC has the best clustering quality in most issues ($\frac{22}{32}$) among all schemes.

**Fig. 2.** The percentage of the issues in cluster 1 and cluster 2.

Experiment Three: Evaluating the scalability by synthetic workload.

Figure 3(a) shows the performance comparison between STC and Basic when the number of transactions $|D|$ varies from 5000 to 20000. In this experiment, the number of items, denoted by N^I , is set to 1000 and the size of the transaction is picked from a Poisson distribution with mean $|T| = 5$. It shows that algorithm STC in general outperforms algorithm Basic in clustering large market-basket database. In addition, it is shown in Figure 3(b) that STC possesses very good scalability, showing another advantage of the self-tuning mechanism. The curve of $SLR(k)$ is the result obtained by SLR with the setting of the fixed SL ratio threshold α to be k .

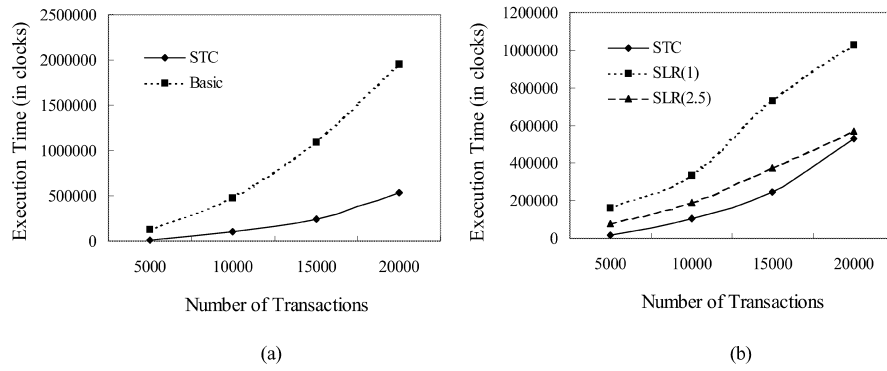


Fig. 3. Scale-up analysis of clustering algorithms.

Experiment Four: When β varies in real data. In this experiment for real data, we set $|D| = 98934$ and $\mu = 5$. To examine the sensitivity of varying the SL distribution rate β , Table 2 shows that STC increases the number of large items in the refinement phase, where I_{C_i, U_0}^L is the number of large items in the initial clustering U_0 and $I_{C_i}^L$ is the number of large items in the final clustering. It can be seen that STC puts similar transactions together by self-tuning SL ratios iteratively.

5 Conclusion

In this paper, we designed algorithm STC for clustering market-basket data items. By utilizing a self-tuning technique for adaptively tuning the input and output SL ratio thresholds, algorithm STC is able to minimize the SL ratios of transactions in the clusters efficiently. We conducted several experiments on the real data and the synthetic workload for performance studies. It is shown by our experimental results that by utilizing the self-tuning technique for adaptively

Table 2. Comparison between the initial clustering and the final clustering on the number of large items for real data when β varies.

	$\beta = 10\%$		$\beta = 20\%$		$\beta = 30\%$		$\beta = 40\%$	
Cluster ID	I_{C_i, U_0}^L	$I_{C_i}^L$	I_{C_i, U_0}^L	$I_{C_i}^L$	I_{C_i, U_0}^L	$I_{C_i}^L$	I_{C_i, U_0}^L	$I_{C_i}^L$
C_1	70	80	173	177	251	253	325	328
C_2	18	60	77	137	120	187	232	291
C_3	41	65	112	138	226	261	303	343
C_4	29	48	70	94	156	211	297	343
C_5	23	49	67	133	163	229	230	303
Average	36.2	60.4	99.8	135.8	183.2	228.2	277.4	321.6

minimizing the input and output SL ratio thresholds, algorithm STC performs very well in both synthetic and real data sets. Specifically, algorithm STC not only incurs an execution time that is significantly smaller than that by prior works but also leads to the clustering results of very good quality.

References

1. UCI Machine Learning Repository. <http://www.ics.uci.edu/~mllearn/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. 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.
4. S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. *ACM SIGMOD International Conference on Management of Data*, 27(2):73–84, June 1998.
5. S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Proceedings of the 15th International Conference on Data Engineering*, 1999.
6. A. K Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computer Surveys*, 31(3), Sept. 1999.
7. K. Wang, C. Xu, and B. Liu. Clustering Transactions Using Large Items. *Proceedings of ACM CIKM International Conference on Information and Knowledge Management*, 1999.
8. Y. Xiao and M. H. Dunham. Interactive Clustering for Transaction Data. *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2001)*, Sept. 2001.
9. C.-H. Yun, K.-T. Chuang, and M.-S. Chen. An Efficient Clustering Algorithm for Market Basket Data Based on Small-Large Ratios. *Proceedings of the 25th International Computer Software and Applications Conference (COMPSAC 2001)*, October 2001.
10. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD International Conference on Management of Data*, 25(2):103–114, June 1996.