# Clustering Categorical Data

Yi Zhang   Ada Wai-chee Fu   Chun Hing Cai   Pheng-Ann Heng
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, Hong Kong
Email: adafu@cse.cuhk.edu.hk

**Abstract**

Clustering has typically been a problem related to numerical data. However, in databases, oftentimes the data values are categorical and cannot be assigned meaningful numerical substitutes. With the recent interest in data mining, we begin to question the possibility of clustering numerical data. Following some recent work in this area, we propose an algorithm based on dynamical systems. To our knowledge, this is the first such algorithm that can guarantee the convergence of the dynamical system, which is a very important property for successful application. We demonstrated the effectiveness of the proposed method on both real data and synthetic data. We also propose a second method based on a graph partitioning approach, for which a new definition of similarity between two nodes is tailored for categorical data.

## 1 Introduction

Mining numerical data has received much attention in recent research in data mining. One important form of knowledge that can be derived from such data is the clustering of the data. Clustering typically groups data into sets in such a way that the intracluster similarity is maximized while the intercluster similarity is minimized [6]. The conventional clustering techniques [15, 7, 4] exploits the inherent geometric properties based on some priori structure for numerical data. There are quite a number of previous works on the clustering problem by the database research community. Some examples are CLARANS [18], DBSCAN [9], DBCLASD [21], Incremental DBSCAN [8], GRIDCLUS [19], CURE [13], K-mean BIRCH [22],

However, much of the data contained in databases is categorical. Much less work has been done on clustering for categorical data. As categorical data usually do not have inherent geometric properties, the clustering of categorical data seems more complicated than that of numerical data. In fact, for numerical data we can easily define the similarity from data's geometric position, whereas for categorical data it is difficult to define what the similarity should be. Here we examine one possible form of similarity for categorical data based on the association via common tuples as well as common categorical values. this meaningful definition of the similarity is given by [12] and it is from an observation of the relational table that contains the categorical data.

The mining of association rules [3, 2] have proven to be an effective data mining technique for basket data. The aim of association rule mining is to find out the items that often appears together in a transaction. That is, if we consider the set of transactions as a 0/1 relational table, association rules look for horizontal co-occurrences of the value 1. We link items (which are the attributes or columns of the table) via their common values in rows. With categorical data, we assume that in the relation, the values are not only 0 or 1 as in the basket database in the binary association rule setting. We assume that each column of the table correspond to an attribute, e.g. a company name, and the values in the column can thus be categorical. In such a case, in addition to horizontal co-occurrences, we can also look at the vertical co-occurrences. That is we can link up rows via their common values for the same attribute. The vertical linkage allows us to group attribute values together in a meaningful way. The result could lead to meaningful clusters.

Let us quote an example given by [12]: suppose we have a database for car sales with the following fields

| manufacturer | model | dealer | price | color | customer | sale date |
|---|---|---|---|---|---|---|
| | | | | | | |

Extracting horizontal correlation in the table allows us to mine rules of the form: of the tuples containing "Honda", 18% contain "August". However, we may also be interested to know that "Honda" and "Toyota" are related by the fact the a large fraction of each are sold in August. Such a fact will not become an association rule since "Honda" and "Toyota" will not appear in the same tuple. However, such a fact can be useful. In fact, we may also relate further that August is a month where a certain dealer has a large volume, so this dealer has some relationship with "Honda" and "Toyota". By grouping values that are linked either vertically or horizontally in the table, we can identify meaningful clusters. [12] proposes a novel approach to solve this problem by a dynamical systems approach, we shall describe this in more details in the next section.

Another previous work that is related to this theme is the clustering of items by hypergraph approach [14]. The aim of this work is to cluster related items in a basket database. For example, for a supermarket, clusters of items can be placed in the same area to facilitate easy access for the customers. The hypergraph is used as an model for the relatedness. A hypergraph is an extension of a graph such that each hyperedge may be identified by more than two nodes. The node set corresponds to distinct items in the database. The frequent itemsets used to derive association rules are used to group items into a hypergraph edge. The weight of a hyperedge is determined by a function of the confidences of all the association rules involving all of the items of the hypergraph. The problem is then how to partition the hypergraph into good clusters. The problem of computing an optimal bisection of a unweighted hypergraph such that the number of hyperedges connecting the partitions is minimized is NP-complete [11]. However many heuristic algorithms have been developed.

A hypergraph partitioning algorithm HMETIS [17] is used in [14] to find the clusters. HMETIS is a multi-level algorithm that produces $k$-way partitions for a weighted hypergraph in such a way that the sum of the weights that straddle partitions is small. The value of $k$ is specified by the user. The algorithm can find good partitions only if the hypergraph is reasonably sparse. The sparseness is achieved by only considering hyperedges (itemsets) with sufficient support. The complexity of HMETIS for a $k$-way partition is $O((V + E)\log k)$ where $V$ is the number of nodes and $E$ is the number of edges.

However, the approach in [14] is targeted for binary transactional data. We are interested to see if similar techniques can be applied to categorical data.

In this paper we show how the known dynamical systems for the above problem cannot guarantee convergence and propose a revised dynamical system and show that it converges. A second method is also proposed based on the graph partitioning approach. This paper is organized in the following way: Section 2 is a description of the dynamical systems approach as proposed in [12]. Sectionsec:dyn introduces our dynamical systems method, and its convergence is shown in Section 4. Section 5 describe how to find clusters in categorical data based on the proposed dynamical systems. The experimental results are presented in Section 6. The second approach based on graph partitioning is presented in Section 7. Section 8 is a conclusion.

## 2 Dynamical Systems Approach

The similarity measure based on the co-occurrence of values in the database is used in [12]. A novel approach is proposed which is based on an iterative method for assigning and propagating weights on the categorical values in a relational table. We can seed a particular node of interest $X$ with some weight. This weight will then propagate to nodes with which $X$ co-occurs frequently. These nodes, having acquired weights, propagate them further. For example, if "Honda" is the seed of interest, the weight will propagate to items which "Honda" co-occurs frequently, such as in the same tuple, via common sale times, common price range, etc.

The approach can be mapped to a certain type of nonlinear dynamical systems. If the dynamical system converges, the categorical database can be clustered. The algorithm can be described as follows. We consider a relational table with $k$ fields (or columns), each of which can assume one of a number of possible values. We represent each possible value in each possible filed by an abstract **node**, and we represent the data as a set $T$ of tuples. Each tuple $\tau \in T$ consists of one node from each field. This corresponds to the hyperedge in the hypergraph setting of the approach in [14]. Let us denote the nodes by $v_1, v_2, ...., v_m$. A **configuration** is an assignment of a weight $w_i$ for each node $v_i$. We also need a *normalization* function $N(w)$ to rescale the weights of the nodes associated with each field so that their squares add up to 1.

We provide an initial configuration and then the procedure shown in Figure 2 can be repeated

To update the configuration $W$:
    create a temporary configuration $W'$
    for each weight $w_s \in W$
        {for each tuple $\tau = \{v_s, v_{i_1}, \cdots, v_{i_{q-1}}\}$ containing $v_s$ do
            $x_\tau \leftarrow \oplus(w_{i_1}, \cdots, w_{i_{q-1}})$.
            $w'_s \leftarrow \sum_\tau x_\tau$.
        }
    $W \leftarrow W'$
    $W \leftarrow N(W)$

Figure 1: procedure for weight propagation

until the configuration does not have further changes.

The combining operator $\oplus$ can be one of a number of choices, including the product operator, the addition operator, etc. The above algorithms actually defined a kind of nonlinear discrete dynamical systems, which can be described as

$$W(k+1) = f(W(k))$$

where $W(k)$ is the configuration in step $k$, and $f$ is the procedure described in the above psuedocode. The algorithm will end if there is a fixed point $W(k)$, for which $W(k+1) = f(W(k)) = W(k)$.

For the initial weights for an initial configuration, we can initialize the weights over a node $v$. We set $w_v = 1$, and $w_u = 1$ for every node $u$ appearing in a tuple with $v$, and $w'_u = 0$ for every other node $u'$. Then $w$ is normalized by $N$. In this way, nodes more "related" to $v$ should acquire larger weights.

The aim of the process is to separate the nodes into two clusters. This can be achieved by a method as follows: We maintain a set of configurations $W_1, ..., W_m$. Initial configurations are given to these as $W_1(0), ..., W_m(0)$. Then we use a standard method to keep the $m$ configurations orthonormal. The standard method used is the Gram-Schmidt procedure [10]. Hence the above process is augmented to be:

(1) $W_i(k+1) = f(W_i(k))$, for $i = 1, 2, ...m$
(2) update the set of vectors $W_i(k+1)$ so that it is orthonormal

Forcing the configurations to remain orthonormal as vectors introduces negative weights into the configuration. From the positive and negative weights in the final configuration, we can

| Tuple | Attribute $a$ | $b$ |
|---|---|---|
| 1. | $A$ | $B$ |
| 2. | $C$ | $D$ |
| 3. | $A$ | $B$ |
| 4. | $C$ | $D$ |

Figure 2: Example

distinguish two separate clusters for the nodes. From experiments, the groups with large positive weights and the nodes with large negative weights tend to represent two meaningful clusters in the nodes.

The dynamical systems approach has several advantages, it has nice mathematical properties and also involves very few arbitrary parameters. Also, as observed in [12], clustering naturally lends itself to combinatorial formulation and the natural combinatorial versions of this problem are NP-complete [11]. The dynamical systems approach is less combinatorial and have been found to be efficient in many contexts. A better pointer to related work can be found in [12].

# 3 The proposed Method

Our proposed method is similar to the method of [12]. One important aspect of using the dynamical systems approach for clustering categorical data is the consideration of the convergence of the dynamical systems obtained by mapping the configuration updating algorithms. If the convergence cannot be guaranteed, then the clustering task may not be able to stop at a solution. In the following we give an example to show that the convergence theorem (Theorem 1) in [12] is not correct. This theorem states that if the combining operator of addition is used, then for every set $T$ of tuples, and every initial configuration $w$, $w$ converges to a fixed point by the proposed mechanism.

Let us consider the simple categorical database as shown in Figure 3. One possible initialization of the weights is to assign weight 1 to A, 0 to C, and equal weights to B and D of $\sqrt{\frac{1}{2}}$. Let us represent the weights of $\{A, B, C, D\}$ by a vector, the initial values is given by $\{1, \sqrt{\frac{1}{2}}, 0, \sqrt{\frac{1}{2}}\}$. After one iteration of the above algorithm this becomes $\{2\sqrt{\frac{1}{2}}, 2, 2\sqrt{\frac{1}{2}}, 0\}$, and this is normalized to $\{\sqrt{\frac{1}{2}}, 1, \sqrt{\frac{1}{2}}, 0\}$. After the second iteration, we obtain $\{2, 2\sqrt{\frac{1}{2}}, 0, 2\sqrt{\frac{1}{2}}\}$, which is normalized to $\{1, \sqrt{\frac{1}{2}}, 0, \sqrt{\frac{1}{2}}\}$. This is the same as the initial configuration. Hence we are back to square one and we have an infinite loop.

To update the configuration $W$:

create temporary configuration $W'$ with weights $w'_1, ..., w'_m$

for each weight $w_{u_i} \in W$

{for each tuple $\tau = \{u_1, u_2, \cdots, u_k\}$ containing $u_i$ do

$x_\tau \leftarrow w_{u_1} + \cdots + cw_{u_i} + \cdots + w_{u_k}$.

$w'_{u_i} \leftarrow \sum_\tau x_\tau$.

}

$W \leftarrow W'$

$W \leftarrow N(W)$

Figure 3: The proposed weight propagation procedure

### 3.1 The revised dynamical systems

In this section, we define a new configuration updating algorithm for clustering categorical datasets, which can guarantee convergence. As we have discussed above, the dynamic algorithm as proposed in [1] may not converge. We amend this problem by modifying the algorithm as shown in Figure 3. In the figure $c$ is a certain positive constant. As we discussed in section 2, the above algorithm define a dynamical system. This algorithm is different from that of [1], since we included the weight of $u_i$ in the $\oplus$ operation to update configuration. This is an important modification since it can ensure the convergence of the dynamical systems.

We also need a *normalization* function to rescale the weights of the nodes. In [12], this normalization is associated with each field, so that the squares of all attribute values for an attribute sum up to 1. However, this will favor fields with fewer attribute values. In other words, an attribute value $A$ that belong to a field with a few values can gain higher weights than values that belong to a field with many different values, even if the value $A$ has less relation to the cluster of interest. Therefore, we normalize the weights of all attribute values so that their squares add up to 1. Let us call the normalization function $N'(W)$.

The dynamical system can be written in compact form as follows.

$$W(k+1) = \frac{1}{\sqrt{[QW(k)]^T[QW(k)]}} QW(k)$$

where

$$Q = \begin{bmatrix} q_{11}c & q_{12} & \dots & q_1n \\ q_{21} & q_{22}c & \dots & q_2n \\ : & : & \dots & : \\ : & : & \dots & : \\ q_{n1} & q_{n2} & \dots & q_n nc \end{bmatrix}$$

$q_{ij}$ $(i, j = 1, ..., n)$ are nonnegative integers. Each $q_{ij}$ is the number of tuples in which $x_i$ and $x_j$ co-occur. $q_{ii}$ is the number of occurrences of $x_i$ in the database. Such a matrix satisfies the following conditions: If $q_{ii} = q_{kk}$ then $\sum_{i=1;j}^{n} q_{ij} = \sum_{j=1;j}^{n} q_{kj}$. If $q_{ii} > q_{kk}$ then $\sum_{i=1;j}^{n} q_{ij} > \sum_{j=1;j}^{n} q_{kj}$ for $j = 1, ..., n$.

The factor of $\frac{1}{\sqrt{[QW(k)]^T[QW(k)]}}$ represents the normalization function $N'(W)$.

**Example**: Consider the following data set

| Tuple | Attribute $a$ | $b$ | $c$ |
|---|---|---|---|
| 1. | $A$ | $R$ | 1 |
| 2. | $A$ | $X$ | 1 |
| 3. | $B$ | $R$ | 2 |
| 4. | $B$ | $X$ | 2 |
| 5. | $C$ | $Y$ | 3 |
| 6. | $C$ | $Z$ | 3 |

Using the configuration updating algorithm we can get a discrete dynamical system as follows.

$$W(k+1) = \frac{1}{\sqrt{[QW(k)]^T[QW(k)]}} QW(k)$$

where $W(k)$ is the vector containing the weights of the attributes values at the $k$-th iteration: $w_A(k)$, $w_B(k)$, ... $w_3(k)$, and

$$Q = \begin{bmatrix} 2c & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 2c & 0 & 1 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 2c & 0 & 0 & 1 & 1 & 0 & 0 & 2 \\ 1 & 1 & 0 & 2c & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 2c & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & c & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & c & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 1 & 0 & 0 & 2c & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 2c & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 2c \end{bmatrix}$$

Let the attribute values be renamed as $A_1 = A, A_2 = B, A_3 = C, A_4 = R, A_5 = X, A_6 = Y, A_7 = Z, A_8 = 1, A_9 = 2, A_{10} = 3$. In the array $A$, the elements $a_{ij}$, $i \neq j$ corresponds to attribute values $A_i$ and $A_j$. The value of $a_{i,j}$, $i \neq j$, is the number of tuples in the database containing both $A_i$ and $A_j$. For example, $A_{14}$ corresponds to $A_1 = A$ and $A_2 = R$, it is equal to

1 since there is exactly 1 tuple containing both $A$ and $R$ in the given database. If the number of tuples that contains $A_i$ is $k$, then the value of $a_{ii}$ is set to $ck$ for a certain constant $c$.

The convergence of this method will be shown in the following section. When the system converges, the items with large weights in the resulting configuration should correspond to a good cluster. We may treat the items with smaller weights as another cluster, or we can repeat the clustering iterations with these items.

In [12], there is a mechanism to introduce negative weights by a orthonormalization of a number of configurations $W_1, ..., W_m$. (see Section 2). It is claimed that the positive and negative weights corresponds to two clusters. However, if the clustering is very obvious, there is a chance that the configurations $W_1, ..., W_m$ will be the same even if they begin with different initial configurations, in which case they cannot become orthonormal. Also we believe that the values of the weights in our approach can reflect the clustering. Therefore, we do not make use of the orthonormalization mechanism.

# 4 Convergence of the proposed algorithm

In this section, we discuss the the convergence of the proposed algorithms. First we describe some properties of some more general dynamical systems of the form that is used by our proposed method. We consider the dynamical systems described by the following systems

$$W(k+1) = f(W(k)) = \frac{1}{\sqrt{[QW(k)]^T[QW(k)]}} QW(k) \tag{1}$$

where $Q$ is a nonzero real symmetric matrix. Furthermore, we assume that all the eigenvalues of $Q$ are positive. We shall see in the following that this assumption is important for convergence.

Since the matrix $Q$ is symmetric, $Q$ is associated with an orthonormal base in $R^n$ and every element of the base is an eigenvector of $Q$. Using this property, we can solve the nonlinear equation (1) in the following.

**Theorem 1** *Suppose that all the eigenvalues $\lambda_1, \cdots, \lambda_n$ of $Q$ are positive and let $S_i (i = 1, \cdots, n)$ be the corresponding eigenvectors with the property that $S_i (i = 1, \cdots, n)$ forms an orthonormal base for $R^n$. For any non zero vector $W(0) \in R^n$, if $W(0) = \sum_{i=1}^n z_i(0) S_i$. Then the solution of (1) starting from $W(0)$ is represented as*

$$W(k) = \sum_{i=1}^n \frac{\lambda_i^k \cdot z_i(0)}{\sqrt{\sum_{j=1}^n \lambda_j^{2k} \cdot z_j^2(0)}} \cdot S_i$$

*for all $k \geq 0$.*

The next theorem shows the convergence of solutions of the dynamical systems.

**Theorem 2** *Suppose that all the eigenvalues of $Q$ are positive, then the solution of (1) starting from any nonzero points in $R^n$ converges to an unit eigenvector of $Q$.*

Next we shall discuss the convergence of the proposed algorithm. The convergence is guaranteed by choosing a suitable constant $c$. By Theorem 2, we known that the convergence of the dynamical system (1) is dependent on the distribution of the eigenvalues of the symmetric matrix $Q$. If all the eigenvalues of $Q$ are positive, then the convergence could be guaranteed. The next question is what kind of matrix $Q$ can satisfy this condition. It is known that all the eigenvalues of a symmetric matrix are positive if and only if this matrix is positive definite. A matrix $A$ is positive definite if $X^TAX > 0$ for all nonzero vector $x$.

$$X^TAX = \sum_{i,j}(a_{ij}x_jx_i)$$

where $a_{ij}$ are the elements of $A$ and $x_i, x_j$ are elements of $X$. Keeping in mind that all elements of the matrix $Q$ above are nonnegative, the value of $a_{ii}x_ix_i$ is positive for any $x_i$. To ensure that a matrix is positive definite, we can make sure that $a_{ii}$ is sufficiently large, for all $i$. If $a_{ii}x_ix_i$ is large enough, the sum in the above will be positive. Therefore, by choosing $c$ suitably large in the above matrix, the convergence of the algorithm can be guaranteed. This property is utilized in the clustering method described in the next section.

## 5 Clustering

In this section, we discuss how the dynamical systems defined above can be used to find clusters for categorical data. We consider the example in Figure 2. It can be mapped to a dynamical system described as

$$X(k+1) = \frac{1}{\sqrt{[QX(k)]^T[QX(k)]}}QX(k) \quad \text{where} \quad Q = \begin{bmatrix} c & 2 & 0 & 0 \\ 2 & c & 0 & 0 \\ 0 & 0 & c & 2 \\ 0 & 0 & 2 & c \end{bmatrix}$$

It is easy to check that the eigenvalues of $A$ are $\lambda_{1,2} = c + 2$ and $\lambda_{3,4} = c - 2$. Taking $c = 3$, then $\lambda_{1,2} = 4$ and $\lambda_{3,4} = 1$. By the convergence theorem, the solution of the dynamical system starting from any point will converge.

Choose an unit initial value as

$$X(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

we have

$$X(1) = \frac{1}{\sqrt{10}} \begin{pmatrix} 3 \\ 1 \\ 0 \\ 0 \end{pmatrix}, X(2) = \frac{1}{2\sqrt{6}} \begin{pmatrix} 5 \\ 3 \\ 0 \\ 0 \end{pmatrix}, X(3) = \frac{1}{\sqrt{130}} \begin{pmatrix} 9 \\ 7 \\ 0 \\ 0 \end{pmatrix} \ldots \longrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Obviously, $\{A, B\}$ and $\{C, D\}$ are clusters.

Therefore, the algorithm of clustering is to start with some value of $c$ for the matrix $Q$, do the iterations in the dynamical system as discussed in Section 3. If the iterations do not converge, repeat the process with a greater value for $c$. This is repeated until the iterations converge.

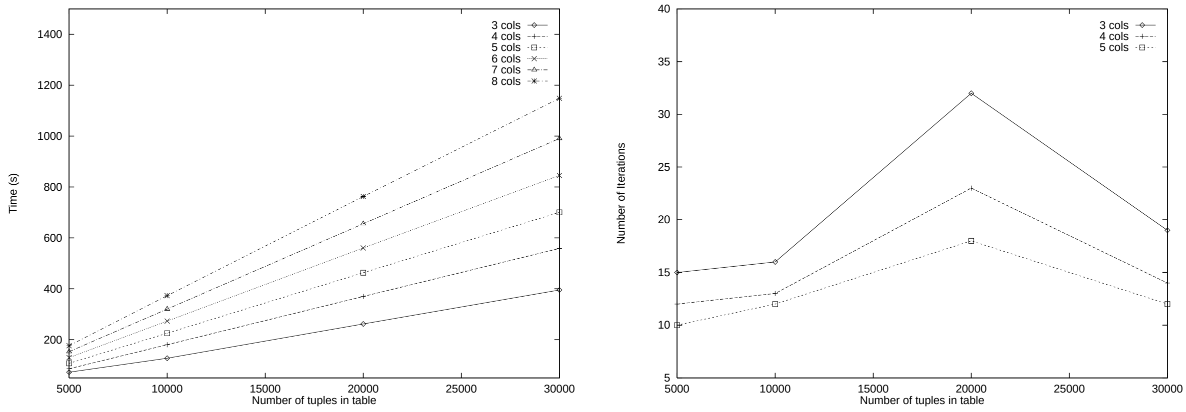
After the dynamical system has converged, we can divide the attribute values in to 2 clusters by grouping the greatest weights and also the lowest weights. Or, we can only group the attribute values with the greatest weights, and then the remaining values will go through the same process. Repeating this process will allow us to get more clusters if necessary.

# 6 Experiments

The experiments were performed on Sun Ultra Sparc 1/170 with 512MB, running on Solaris 2.6. Programs for the experiments are written in C. Two different types of data are used. The first type is synthetic data. It is a randomly-generated categorical table in which we plant extra random tuples involving only a small number of nodes in each column. This subset of nodes will be the expected cluster since they co-occur frequently compared to the case of a random table. This way of data generation follows that of [12], which has been used as early as [5]. It is called quasi-random input, with carefully planted structure among random noise. The second set is real data from a bibliographical database. In all the experiments the iterations stop when the change in the weight of each attribute value is less than 0.001.

## 6.1 Performance

For the running time, an experiment is carried out with synthetic data. The algorithm scales well with increasing data volumes. We measured the running time for 10 iterations, for varying numbers of tuples and number of columns. The results are shown in Figure 6.1(a). We can see the running time for the same number of iterations is linear in the number of tuples and nearly linear in the number of columns. We also measure the number of iterations required for the system to stabilize. The results are shown in Figure 6.1 (b). The number of iterations required ranges from 10 to 35. The efficiency is quite acceptable.




(a) (b)

Figure 4: Performace

## 6.2 Accuracy

As in [12], we define a measurement of the accuracy called purity. First we examine how well the system works for data with a single expected cluster. In a categorical data with 3 columns, there are 1000 distinct attributes values per column, 5000 rows are generated randomly. Extra rows are planted using only attributes values from a set of 30 (10 from each column), let us call these attribute values the *clustering values*. When the dynamic systems has stopped, we examine the attribute values with the top 10 weights, **purity** is the percentage of these top attribute values coming from the clustering values. The results are shown in Figure 5 (a). We can see that a smaller number of extra rows would introduce more inaccuracy, since the pattern can be more easily blurred by random data. However, when the extra rows reaches 75 or above, the result becomes very accurate.

The next set of experiment measures how accurate the system is for data with two embedded clusters. For this we create a categorical data table with 3 columns, there are 1000 distinct attribute values per column, again 5000 rows are generated randomly. Two distinct sets of 30 attribute values each are identified. Two sets of extra rows are now added with values drawn from the two sets of values respectively. Hence the two sets of values correspond to two expected clusters. When the dynamic systems has stopped, we examine the attribute values with the top 10 weights, **purity** is the percentage of these top attribute values coming from exactly one of the two expected clusters. Note that this measurement is different from the one used in [12] since our clustering algorithm is different: we do not look at both ends of the weight distribution. The results are shown in Figure 5 (b).

Note that the two clusters are most likely not disjoint, since the random tuples can provide linkage among attribute values of the two clusters. Once two clusters are not disjoint, the weights

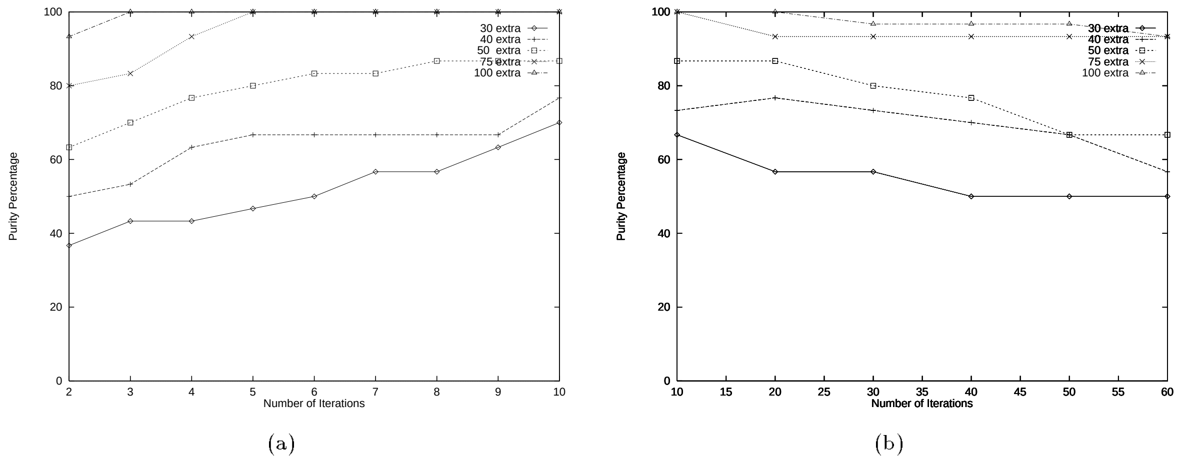



Figure 5: Purity

of one cluster can "leak" through the linkage to the other cluster. This leakage may not happen immediately as the weight propagation may take time. This is the reason why we see a deterioration in the purity for increasing number of iterations. However, the purity will still be quite high when the system stabilize.

## 6.3 Bibliographical data

The system of [12] is called STIRR. Although STIRR does not guarantee convergence it is a good benchmarking to compare our method with STIRR. Experiments are conducted on real bibliographical data as used in [12]. This is a publicly-accessible bibliographic databases with 4700 papers from database research and 66,000 papers written on theoretical computer science and related fields, two data sets are constructed each with four columns. For each paper, we recorded the name of the first author the name of the second author (if any), the conference or journal of publication, and the year of publication. Thus, each paper became a relation of the form (Author-1, Author-2, Conference/Journal, Year). In addition to the tables of theory papers and database papers, we also constructed a mixed table with both sets of papers. The results are shown in Figure 6. The seed weight is planted at attribute value of "VLDB". The STIRR algorihtm requires 8 iterations and the propsed method requires 16 iterations. We can see that the results are quite similar.

Since the above set up consists of a much greater proportion of theoretical paper, even if we plant a seed at the database related paper, the cluster that comes out is still the theoretical papers. Hence we examine the case where the number of theoretical paper is close to that of the number of database research paper. We also select randomly 6000 records from the theoretical papers database, and 4000 records from the database papers database. The results are shown

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1990 | 0.397 | LIBTR | 0.682 | Chen | 0.275 | Chen | 0.228 |
| 1992 | 0.373 | IPL | 0.340 | Chang | 0.142 | Lee | 0.194 |
| 1991 | 0.369 | TCS | 0.330 | Wang | 0.141 | Sharir | 0.163 |
| 1993 | 0.332 | TR | 0.192 | Lee | 0.137 | Wang | 0.163 |
| 1994 | 0.325 | SICOMP | 0.130 | Li | 0.136 | Naor | 0.114 |
| 1989 | 0.281 | FOCS | 0.128 | Agrawal | 0.129 | Miller | 0.111 |
| 1995 | 0.269 | LNCS | 0.123 | Alon | 0.126 | Li | 0.109 |
| 1996 | 0.215 | DAMATH | 0.123 | Lin | 0.120 | Tarjan | 0.109 |

Results of the STIRR algoirthm for skewed clusters

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1992 | 0.467 | LIBTR | 0.902 | Chen | 0.501 | Chen | 0.364 |
| 1990 | 0.437 | IPL | 0.281 | Chang | 0.164 | Lee | 0.0261 |
| 1991 | 0.423 | TCS | 0.228 | Wang | 0.159 | Sharir | 0.211 |
| 1994 | 0.344 | TR | 0.091 | Agrawal | 0.158 | Wang | 0.190 |
| 1993 | 0.336 | SICOMP | 0.062 | Lee | 0.153 | Naor | 0.123 |
| 1995 | 0.257 | FOCS | 0.061 | Li | 0.150 | Vitter | 0.119 |
| 1989 | 0.214 | STOC | 0.055 | Alon | 0.130 | Li | 0.117 |
| 1996 | 0.176 | DAMATH | 0.051 | Lin | 0.127 | Tarjan | 0.113 |

Results of the proposed algoirthm for skewed clusters

Figure 6: Results for Skewed clusters

in Figure 7. The seed weight is planted at attribute value of "VLDB". The STIRR algorihtm requires 10 iterations and the propsed method requires 20 iterations. We can see that the results are quite similar. This time, we discovered the database cluster.

# 7 The Second Proposed Method

The second proposed method is related to the hypergraph segmentation as proposed in [14]. However, since we are dealing with categorical data, we need to give the edges a new definition of weights. In [14], each edge of the hypergraph corresponds to an itemset $\{A_1, A_2, ..., A_k\}$, where each $A_i$ is an attribute value. The weight is given by a function of the confidence of the underlying association rules. For example, for an edge for $\{A_1, A_2\}$, the weight can be the average of $P(A_1|A_2)$ and $P(A_2|A_1)$. This definition is fine for the basket data that [14] considers, in which the relational table contains only binary information. However, we cannot adopt the definition for the categorical database. Consider two edges, one for $\{A_1, A_2\}$, and one for $\{A_3, A_2\}$, where $A_1$ is a categorical value for attribute $a_1$, $A_2$ is a categorical value for attribute $a_2$, and $A_3$ is a categorical value for attribute $a_3$. Suppose there are only two categorical values $a_1$, but there are 100 different categorical values for $a_3$. Then even if there is no particular correlation between

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1990 | 0.434 | LIBTR | 0.618 | Chen | 0.242 | DeWitt | 0.162 |
| 1991 | 0.375 | ICDE | 0.285 | Wiederhold | 0.209 | Lee | 0.144 |
| 1992 | 0.370 | IPL | 0.270 | Yu | 0.280 | Kim | 0.136 |
| 1993 | 0.330 | VLDB | 0.244 | Abiteboul | 0.130 | Wang | 0.129 |
| 1989 | 0.288 | TCS | 0.212 | GarciaMolina | 0.129 | Wiederhold | 0.122 |
| 1994 | 0.236 | SIGMOD | 0.211 | Litwin | 0.128 | Chang | 0.108 |
| 1988 | 0.214 | ACMTDS | 0.136 | Huang | 0.124 | Park | 0.107 |
| 1987 | 0.199 | PODS | 0.127 | Martin | 0.108 | Wong | 0.103 |

Results of the STIRR algoirthm for even clusters

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1990 | 0.662 | LIBTR | 0.900 | Chen | 0.439 | DeWitt | 0.222 |
| 1991 | 0.437 | ICDE | 0.196 | Wiederhold | 0.354 | Lee | 0.198 |
| 1992 | 0.373 | IPL | 0.178 | Yu | 0.256 | Kim | 0.160 |
| 1993 | 0.268 | VLDB | 0.177 | GarciaMolina | 0.130 | Wang | 0.157 |
| 1989 | 0.262 | SIGMOD | 0.138 | Abiteboul | 0.129 | Wiederhold | 0.152 |
| 1994 | 0.150 | TCS | 0.110 | Huang | 0.121 | Park | 0.136 |
| 1988 | 0.126 | ACMTDS | 0.075 | Litwin | 0.116 | Langston | 0.135 |
| 1987 | 0.115 | PODS | 0.066 | Paterson | 0.105 | Shamir | 0.129 |

Results of the proposed algoirthm for even clusters

Figure 7: Results for even clusters

$A_1$ and $A_2$ and there is some correlation between $A_2$ and $A_3$, the value of $P(A_1|A_2)$ could still be greater then that of $P(A_3|A_2)$, simply because there are much fewer alternative values for $a_1$ compared to $a_3$. Hence we need to find another measurement of the weight for the edges.

Suppose there are two attributes $a$ and $b$. $A$ is an attribute value for $a$ and $B$ is an attribute value for $b$. Let $P_A$ be the fraction of tuples containing $A$ in attribute $a$. Let $P_B$ be the fraction of tuples containing $B$ in attribute $b$. If $A$ has no special relationship with $B$, then we expect that the number of tuples containing both $A$ and $B$ to be $P_A \times P_B$. We use the deviation from this expectation to estimate the relationship between $A$ and $B$. Hence if $P_{AB}$ is the actual fraction of tuples containing both $A$ and $B$,

$$linkage_{AB} = \frac{P_{AB} - P_A P_B}{P_A P_B}$$

This definition has resemblance to the definition of $\chi^2$ value in statistics. However, the $\chi^2$ value will be large for both positive correlation and negative correlation. For our purpose, we want to sort all correlations, so that the highest positive correlations will come on top. Another measurement that is related is the mutual information $I(X;Y)$ used in information theory.

We shall only consider a graph with edges for two nodes, that is, we shall not consider hyperedge that may contain three or more nodes. We believe that the relationships for pairs of categorical values are sufficient for the clustering purpose. If three values are actually correlation, we can discover the correlation by means of two linkages with a common node. This is similar to the idea of the weight propagation in the dynamical system, which propagate weights via common categorical values.

## 7.1 The support threshold

The above definition for linkage seems quite reasonable. However let us consider a special case. Suppose we have 5 attributes a,b,c,d,e, there is a tuple {A,B,C,D,E} and $A$ $B$, $C$, $D$ and $E$ appears only once. $P_{AB} = P_A = P_B = \frac{1/N - 1/N^2}{1/N^2} = N - 1$. We can see that the value for $linkage_{AB}$ will be very large, and rightly so, since whenever $A$ appears, the only possible value for $b$ is $B$ and vice versa. However, we would not be very interested in the relationship between $A$ and $B$ because the occurrences of $A$ and $B$ are so rare. This reminds us of the usage of the support value in the association rule mining.

We see a need in a measurement similar to the support of an itemset in the association rule mining. The remaining question is how to set the support threshold. We have the choice of incorporating the value of support into the measurement of $linkage_{AB}$. We also have the choice of first eliminating elements with insufficient support before the consideration of the linkage values. We choose the later method since this approach is also used in the apriori algorithm and is quite successful, and also it allows us to do some pruning to reduce the amount of computations required.

## 7.2 The algorithm

The clustering algorithm is a variant of the single link cluster method [20], which has also been used in [1] for clustering items in a market basket database. We scan the database tuples once. With the scanning, we compute the number of tuples that contain each attribute value, and we compute for each pair of attribute values the number of tuples that contain both values. This will give us the values of $P_A$ and well as $P_{AB}$ for any attribute value $A$ and any pair of attribute values $A, B$. We can compute the linkage $linkage_{AB}$. The greater the value of $linkage_{AB}$, the shorter the distance between the values $A$ and $B$. Initially we assume each attribute value belong to a unique cluster. We shall try to form bigger clusters by iteratively joining clusters. The joining is done by making use of the linkage of the greatest value in the linkage list. Suppose the linkage is for the pair of values $A$ and $B$. Then if $A$ and $B$ belongs to two different clusters $C_1, C_2$, we merge the two clusters into one. The pair $\{A, B\}$ is remove from the linkage list. We repeat the process with the top element remaining in the linkage list. This is repeated until we have exactly two clusters left.

The above mechanism requires only one scan of the entire database. Suppose there are $N$

different attribute values. The number of clusters is at most $N$. Each attribute value will go through at most $N$ changes in its assignment to different clusters, which occurs in the merging step. Hence the complexity is $O(N^2)$. Moreover, if we are only interested in the most correlated values as in the dynamical systems scenarios, we can stop the clustering process at some point and extract the cluster with the strongest linkage. It may also be meaningful to discover a partition of the graph created above in terms of a minimum cut. However, the complexity for this problem is quite high. The most efficient algorithms known to us requires $O(n^2 \log^3 n)$ time, where $n$ is the number of nodes [16].

We have tried this method on small scale experiments and obtained expected clusters. For the bibliographical database, we obtained one very large cluster for the theoretical papers. However, we also discovered odd effects from data with rare occurrences, which may be considered as outliers. We believe that the linkage definition may need to be adjusted for this reason. Further work will be needed for this method.

# 8 Conclusion

Clustering categorical data is an important problem in data mining, for which investigation is still relatively rare. This paper is a study of this problem. We adopt a definition of similarity for categorical data based on common attribute values for relational tuples. A promising approach by dynamical systems is investigated and we propose what we believe to be the first dynamical system for this problem that can gaurantee convergence. We demonstrated the performance of the proposed algorithm in computation efficiency and also effectiveness in clustering by experiments on both real and synthetic data. A second mechanism based on the idea of graph partitioning is also introduced, for which we define weights of the edges based on the categorical nature of the data.

# 9 Appendix: Proofs of the theorems

**Theorem 1**: *Suppose that all the eigenvalues $\lambda_1, \cdots, \lambda_n$ of $Q$ are positive and let $S_i (i = 1, \cdots, n)$ be the corresponding eigenvectors with the property that $S_i (i = 1, \cdots, n)$ forms an orthonormal base for $R^n$. For any non zero vector $W(0) \in R^n$, if $W(0) = \sum_{i=1}^{n} z_i(0) S_i$. Then the solution of (1) starting from $W(0)$ is represented as*

$$W(k) = \sum_{i=1}^{n} \frac{\lambda_i^k \cdot z_i(0)}{\sqrt{\sum_{j=1}^{n} \lambda_j^{2k} \cdot z_j^2(0)}} \cdot S_i$$

*for all $k \geq 0$.*

Proof: For any $k \geq 0$, let $W(k)$ be the solution of (1) starting from $W(0)$. Since $S_i (i = 1, \cdots, n)$ form an orthonormal base of $R^n$, $W(k)$ can be represented as

$$W(k) = \sum_{i=1}^{n} z_i(k) S_i. \tag{2}$$

where $z_i(k)(i = 1, \cdots, n)$ are some real discrete functions. Then, we have

$$W(k+1) = \sum_{i=1}^{n} z_i(k+1) S_i$$

and

$$QW(k) = \sum_{i=1}^{n} \lambda_i z_i(k) S_i$$

and

$$\sqrt{[QW(k)]^T [QW(k)]} = \sqrt{\sum_{i=1}^{n} \lambda_i^2 z_i^2(k)}.$$

It follows from (1) that

$$z_i(k+1) = \frac{1}{\sqrt{\sum_{j=1}^{n} \lambda_j^2 z_j^2(k)}} \lambda_i z_i(k), (i = 1, \cdots, n). \tag{3}$$

Since $\lambda_i > 0 (i = 1, \cdots, n)$, then $z_i(k)(i = 1, \cdots, n)$ have the same sign as $z_i(0)$ for all $k \geq 0$. By the assumption that $W(0) \neq 0$, there must exist an $r(1 \leq r \leq n)$ such that $z_r(0) \neq 0$. Without loss of generality we assume that $z_r(0) > 0$. Then we have

$$\frac{z_i(k+1)}{z_r(k+1)} = \frac{\lambda_i}{\lambda_r} \cdot \frac{z_i(k)}{z_r(k)}, (i = 1, \cdots, n)$$

for all $k \geq 0$. It is easy to see that

$$\frac{z_i(k)}{z_r(k)} = \left(\frac{\lambda_i}{\lambda_r}\right)^k \cdot \frac{z_i(0)}{z_r(0)}, (i = 1, \cdots, n) \tag{4}$$

for all $k \geq 1$. Then, we have

$$z_i(k) = \left(\frac{\lambda_i}{\lambda_r}\right)^k \cdot \frac{z_i(0)}{z_r(0)} \cdot z_r(k), (i = 1, \cdots, n) \tag{5}$$

for all $k \geq 1$.

Next, we find $z_r(k)$. From (3), it follows that

$$z_r(k+1) \quad = \quad \frac{1}{\sqrt{\sum_{j=1}^{n} \lambda_j^2 \cdot z_j^2(k)}} \lambda_r z_r(k)$$

$$
\begin{aligned}
&= \frac{\lambda_r}{\sqrt{\sum_{j=1}^n \lambda_j^2 \cdot \left[\frac{z_j(k)}{z_r(k)}\right]^2}} \\
&= \frac{\lambda_r}{\sqrt{\sum_{j=1}^n \lambda_j^2 \cdot \left(\frac{\lambda_j}{\lambda_r}\right)^{2k} \cdot \left[\frac{z_j(k)}{z_r(k)}\right]^2}} \\
&= \frac{\lambda_r^{(k+1)} \cdot z_r(0)}{\sqrt{\sum_{j=1}^n \lambda_j^{2(k+1)} \cdot z_j^2(0)}} \qquad (6)
\end{aligned}
$$

for all $k \geq 1$.

Therefore,

$$
z_i(k) = \frac{\lambda_i^k \cdot z_i(0)}{\sqrt{\sum_{j=1}^n \lambda_j^{2k} \cdot z_j^2(0)}}, (i = 1, \cdots, n).
$$

Then, from (2) we have

$$
W(k) = \sum_{i=1}^n \frac{\lambda_i^k \cdot z_i(0)}{\sqrt{\sum_{j=1}^n \lambda_j^{2k} \cdot z_j^2(0)}} S_j
$$

for all $k \geq 0$. The proof is completed. □

**Theorem 2**: *Suppose that all the eigenvalues of $Q$ are positive, then each solution of (1) starting from any nonzero points in $R^n$ converges to an unit eigenvector of $Q$.*

Proof: Let $\lambda_i (i = 1, \cdots, n)$ be all the eigenvalues of $Q$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0$. Since $Q$ is a symmetric matrix, there exit a orthonormal base $S_i(i = 1, \cdot, n)$ in $R^n$ and $S_i(i = 1, \cdot, n)$ are eigenvectors of $Q$ corresponding to the eigenvalues $\lambda_i(i = 1, \cdots, n)$. Let $\sigma_i(i = 1, \cdots, m)$ be all the distinct eigenvalues of $Q$ and ordered with $\sigma_1 > \sigma_2 > \cdots > \sigma_m$. For any $i, (1 \leq i \leq m)$, denote the sum of the algebraic multiplicity of $\sigma_1, \cdots, \sigma_i$ by $c_i(1 \leq i \leq n)$. Obviously, we have $c_m = n$. For convenience, we denote $c_0 = 1$. It is easy to see that $\lambda_i \equiv \sigma_r$ for all $i \in [c_{r-1}, c_r)$, and all $S_i(c_{r-1} \leq i \leq c_r)$ belong to the same eigenspace that corresponds to the eigenvalue $\sigma_r$.

Let $W(0)$ be any of a nonzero point in $R^n$, then $W(0)$ can be represented as

$$
W(0) = \sum_{i=1}^n z_i(0) S_i.
$$

Let

$$
l = \min\{l | z_l \neq 0, 1 \leq l \leq n\}
$$

then there exist an $r \in \{1, 2, \cdots, m\}$ such that $c_{r-1} \leq l < c_r$. From (1) we have

$$
W(k) = \frac{1}{\sqrt{\sum_{j=1}^n \lambda_j^{2k} \cdot z_j^2(0)}} \sum_{i=1}^n \lambda_i^k \cdot z_i(0) S_i
$$

$$
\begin{aligned}
&= \frac{1}{\sqrt{\sum_{s=1}^{m} \sigma_s^{2k} \cdot \sum_{j=c_{s-1}}^{c_s} z_j^2(0)}} \sum_{s=1}^{m} \sigma_s^k \cdot \sum_{j=c_{s-1}}^{c_s} z_j(0) S_j \\
&= \frac{\sum_{s=1}^{m} \sigma_s^k \cdot \sum_{j=c_s}^{c_s} z_j(0) S_j}{\sqrt{\sigma_r^{2k} \sum_{j=c_{r-1}}^{c_r} z_j^2(0) + \sum_{s=1, s\neq r}^{m} \sum_{j=c_{s-1}}^{c_s} \sigma_s^{2k} \cdot z_j^2(0)}} \\
&= \frac{1}{\sqrt{\sum_{j=c_{r-1}}^{c_r} z_j^2(0) + \sum_{j=c_{r+1}}^{n} \left(\frac{\lambda_j}{\sigma_r}\right)^{2k} \cdot z_j^2(0)}} \sum_{i=c_{r-1}}^{c_r} z_i(0) S_i \\
&+ \sum_{i=c_r+1}^{n} \frac{1}{\sqrt{\sum_{j=c_{r-1}}^{c_r} z_j^2(0) \cdot \left(\frac{\sigma_r}{\lambda_i}\right)^{2k} + \sum_{j=c_r+1}^{n} z_j^2(0) \cdot \left(\frac{\lambda_j}{\lambda_i}\right)^{2k}}} z_i(0) S_i
\end{aligned}
$$

and so

$$
W(k) \longrightarrow \sum_{i=c_{r-1}}^{c_r} \frac{z_i(0)}{\sqrt{\sum_{j=c_{r-1}}^{c_r} z_j^2(0)}} \cdot S_i
$$

as $k \to +\infty$. This completes the proof. □

## References


[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A new method for similarity indexing of market basket data. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1999.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, 1994.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD, Washington, DC, USA*, pages 207–216, 1993.

[4] M.S. Aldenderfer and R. K. Blashfield. *Cluster Analysis*. Sage, 1984.

[5] R. Boppana. Eigenvalues and graph bisection. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1987.

[6] M.S. Chen, J. Han, and P.S. Yu. Data mining: An overview from database perspective. In *IEEE Transactions on Knowledge and Data Engineering, Vol 5, No. 1, Feb*, pages 866–883, December 1997.

[7] edited by J. Van Ryzin. *Classification and Clustering*. Academic Press, Inc., 1977.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th VLDB Conference, New York, USA*, 1998.

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings International Conference on Knowledge Discovery and Data Mining KDD-98, AAAI Press*, pages 226–231, 1996.

[10] C.F. Van Loan G. Golub. *Matrix Computations.* John Hopkins University Press, 1989.

[11] M.R. Garey and D.S. Johnson. *Computers and Intractability.* Freeman and Company, 1979.

[12] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *Proceedings of the 24th VLDB Conference, New York, USA*, 1998.

[13] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada*, June 1996.

[14] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[15] J. A. Hartigan. *Clustering Algorithms.* John Wiley & Sons, 1975.

[16] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, July 1997.

[17] G. Karypis, R. Aggrawal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Proceedings ACM/IEEE Design Automation Conference*, 1997.

[18] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference, Santiago, Chile*, 1994.

[19] Erich Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *Proceedings of Internation Conference on Pattern Recognition (ICPR)*, pages 101–105, 1996.

[20] R. Sibson. Slink: An optimally efficient algorithm for the single link cluster method. In *Computer Journal*, pages 30–34, 1973.

[21] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings on 14th International Conference on Data Engineering (ICDE'98)*, 1998.

[22] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada*, pages 103–114, June 1996.