Clustering Categorical Data: An Approach Based on Dynamical Systems*

David Gibson
Dept. of Computer Science
UC Berkeley
Berkeley, CA 94720 USA
dag@cs.berkeley.edu

Jon Kleinberg
Dept. of Computer Science
Cornell University
Ithaca, NY 14853 USA
kleinber@cs.cornell.edu

Prabhakar Raghavan
Almaden Research Center
IBM
San Jose, CA 95120 USA
pragh@almaden.ibm.com

Abstract

We describe a novel approach for clustering collections of sets, and its application to the analysis and mining of categorical data. By "categorical data," we mean tables with fields that cannot be naturally ordered by a metric — e.g., the names of producers of automobiles, or the names of products offered by a manufacturer. Our approach is based on an iterative method for assigning and propagating weights on the categorical values in a table; this facilitates a type of similarity measure arising from the co-occurrence of values in the dataset. Our techniques can be studied analytically in terms of certain types of non-linear dynamical systems. We discuss experiments on a variety of tables of synthetic and real data; we find that our iterative methods converge quickly to prominently correlated values of various categorical fields.

1 Introduction

Much of the data in databases is *categorical*: fields in tables whose attributes cannot naturally be ordered as numerical values can. The problem of *clustering* categorical data involves complexity not encountered in the corresponding problem for numerical data, since one has much less *a priori* structure to work with. While considerable research has been done on clustering numerical data (exploiting its inherent geometric properties), there has been much

*A version of this paper appears in the Proceedings of the 24th International Conference on Very Large Databases, 1998.

less work on the important problem of clustering and extracting structure from categorical data.

As a concrete example, consider a database describing car sales with fields "manufacturer", "model", "dealer", "price", "color", "customer" and "sale date". In our setting, price and sale date are traditional "numerical" values. Color is arguably a categorical attribute, assuming that values such as "red" and "green" cannot easily be ordered linearly; more on this below. Attributes such as manufacturer, model and dealer are indisputably categorical attributes: it is very hard to reason that one dealer is "like" or "unlike" another in the way one can reason about numbers, and hence new methods are needed to search for similarities in such data.

In this paper we propose a new approach for clustering categorical data, differing from previous techniques in several fundamental respects. First, our approach generalizes the powerful methodology of *spectral graph partitioning* [16, 19] to produce a clustering technique applicable to arbitrary collections of sets. For a variety of graph decomposition tasks, spectral partitioning has been the method of choice for avoiding the complexity of more combinatorial approaches (see below); thus one of our main contributions is to generalize and extend this framework to the analysis of relational data more complex than graphs, obtaining a method that does not suffer from the combinatorial explosion encountered in existing approaches. Second, our development of this technique reveals a novel connection between tables of categorical data and non-linear dynamical systems; this connection allows for the natural development of a range of clustering algorithms that are mathematically clean and involve essentially no arbitrary parameters.

In addition to the development of the techniques themselves, we report on their incorporation into an experimental system for mining tables of categorical data. We have found the techniques to be effective in the context of both synthetic and real datasets.

We now review some of the key features of our approach, and explain the ways in which it differs from other approaches.

(1) No a priori quantization: We wish to extract notions of proximity and separation from the items in a categorical dataset purely through their patterns of co-occurence, without trying to impose an artificial linear order or numerical structure on them. The approach of casting categorical values into numbers or vectors can be effective in some cases (for one example, see the work on image processing and color perception in [11, 18, 20]; in the simplest form, each color is a 3-d vector of RGB intensities). However, it has a significant drawback: in engineering a suitable numerical representation of a categorical attribute, one may lose the structure one is hoping to mine. Moreover, it can lead to problems in high-dimensional geometric spaces in which the data is distributed very sparsely (as, for example, in the interesting *k-modes* algorithm for categorical data, due to Huang [25]).

(2) Correlation vs. categorical similarity: Association rules and their generalizations have proved to be effective at mining that involves estimating/comparing moments and conditional probabilities; see e.g. the work on forming associations from market basket data in [1, 2, 8, 37]. Our analysis of co-occurrence in categorical data addresses a notion that is in several respects distinct from the underlying motivation for association rules. First, we wish to define a notion of similarity among items of a database that will apply even to items that

never occur together in a tuple; rather, their similarity will be based on the fact that the *sets* of items with which they do co-occur have large overlap. Second, we would like this notion of similarity to be transitive in a limited way: if A and B are deemed similar; and B and C are deemed similar, then we should be able to infer a "weak" type of similarity between A and C. In this way, similarity can propagate to uncover more distant correlations.

The first of these points also serves as part of the underlying motivation of recent independent work of Das, Mannila, and Ronkainen [13]. However, the way in which we will use this notion for clustering is quite different.

(3) New methods for hypergraph clustering: Viewing each tuple in the database as a *set* of items, we can treat the entire collection of tuples as an abstract *set system*, or *hypergraph* [5], and approach the clustering problem in this context. (By a *hypergraph*, in this setting, we mean simply an arbitrary collection of sets.) In this way, we use pure *co-occurrence* information among items to guide the clustering, without the imposition of additional prior structure. At a high level, this is similar to the approach taken by Han, Karypis, Kumar, and Mobasher [23], as well as by the framework of association rules [2].

Clustering a collection of sets is a task that naturally lends itself to combinatorial formulations; unfortunately, the natural combinatorial versions of this problem are NP-complete and apparently difficult to approximate [21]. This becomes a major obstacle in the approach of Han et al. [23] (and to a lesser degree in Das et al. [13]), who rely on combinatorial formulations of the clustering problem and heuristics whose behavior is difficult to reason about.

We adopt a fundamentally different, less combinatorial, approach to the problem of clustering sets; it is motivated by *spectral graph partitioning*, a powerful method for the related problem of clustering undirected graphs that originated in work of Donath and Hoffman [16] and Fiedler [19] in the 1970's. Spectral methods relate good "partitions" of an undirected graph to the eigenvalues and eigenvectors of certain matrices derived from the graph. These methods have been found to exhibit good performance in many contexts that are difficult to attack by purely combinatorial means (see e.g. [4, 12, 26, 36]), and they has been successfully applied in areas such as finite-element mesh decomposition [36] and the analysis of Monte Carlo simulation methods [26]. Recently the heuristic intuition underlying spectral partitioning has been used in the context of information retrieval [14] and in methods for identifying densely connected hypertextual regions of the World Wide Web [9, 28].

Our new method can be viewed as a generalization of spectral partitioning techniques to the problem of *hypergraph clustering*. The extension to this more general setting involves significant changes in the algorithmic techniques required; in particular, the use of eigenvectors is replaced, in our method, by certain types of *non-linear dynamical systems*. One of our contributions here is the generalization of the notion of non-principal eigenvectors (from linear maps) to our nonlinear dynamical systems. In what follows, we will argue that this introduction of dynamical systems is perhaps the most natural way to extend the power of spectral methods to the problem of clustering collections of sets; and we will show that this approach suggests a framework for analyzing co-occurrence in categorical datasets in a way that avoids many of the pitfalls encountered with intractable combinatorial formulations of the problem.

We now turn to an extended example that illustrates more concretely the type of problem addressed by our methodology.

Example and Overview of Techniques. Consider again our hypothetical database of car sales. Association rules are effective at mining patterns of the form: of the tuples containing "Honda," 18% (a large fraction) contain "August." Our goal, on the other hand, is to elicit statements of the form "Hondas and Toyotas are 'related' by the fact that a disproportionate fraction of each are sold in the month of August". Thus, we are grouping "Honda" and "Toyota" by co-occurrence with a common value or set of values ("August" in this case), although clearly in a database of individual car purchases, no single tuple is likely to contain both "Honda" and "Toyota." We wish to allow such groupings to grow in complex ways; for example, we may discover that many Toyotas are also sold in September, as are many Nissans; that many dealers sell both Hondas and Acuras; that many other dealers have large volume in August and September. In this way we relate items in the database that have no direct co-occurrence — different automobile manufacturers, or dealers who may sell different makes of automobile. What emerges is a high-level grouping of database entries; one might intuitively picture it as being centered around the notion of "late summer sales on Japanese model cars." Distilling such a notion is likely to be of great value in customer segmentation, commonly viewed as one of the most successful applications of data mining to date. Note that in general, a dataset could contain many such segments; indeed, our method is effective at uncovering this multiplicity (see Theorem 4 and the experiments in Section 5).

Thus, if we picture a database consisting of rows of tuples, association rules are based on co-occurrence within rows (tuples), while we are seeking a type of similarity based on co-occurrence patterns of different items in the same column (field). Our problem, then, is how to define such a notion of similarity, allowing a "limited" amount of transitivity, in a manner that is efficient, algorithmically clean, and robust in a variety of settings. We propose a weight-propagation method which works roughly as follows. We first seed a particular item of interest (e.g. "Honda") with a small amount of weight. This weight then propagates to items with which Honda co-occurs frequently — dealers, common sale times, common price ranges. These items, having acquired weight, propagate it further — back to other automobile manufacturers, perhaps — and the process iterates. In this way, we can achieve our two-fold objective: items highly related to Honda acquire weight, even without direct co-occurrence; and since the weight "diffuses" as it spreads through the database, we can achieve our "limited" form of transitivity.

The weight-propagation schemes that we work with can be viewed as a type of *non-linear dynamical system* derived from the table of categorical data. Thus, our work demonstrates a natural transformation from categorical datasets to such dynamical systems that facilitates a new approach to the clustering and mining of this type of data. Much of the analysis of such dynamical systems is beyond the reach of current mathematics [35]. In a later section, we discuss some preliminary mathematical results that we are able to prove about our method, including convergence properties in some cases and an interpretation of a more complex formulation in terms of the enumeration of certain types of labeled trees. We also indicate certain aspects of the analysis that appear to be quite difficult, given the current state of knowledge concerning non-linear dynamical systems. It is important to note that even for the systems that cannot be fully analyzed at a mathematical level, we find experimentally that they exhibit quite orderly behavior and consistently elicit meaningful structure implicit in tables of categorical data, typically in linear time.

Organization of the Paper. In the following section, we describe in detail the algorithmic components underlying our approach, and the way in which they can be used for clustering and mining categorical data. We also discuss there the mathematical properties of the algorithms that we have been able to prove.

In Section 3 we describe STIRR (Sieving Through Iterated Relational Reinforcement), an experimental system for studying the use of this technique for analyzing categorical tables. Because the iterative weight-propagation algorithms underlying STIRR converge in a small number of iterations in practice, the total computational effort in analyzing a table by our methods is typically linear in the size of the table. By providing quantitative measures of similarity among items, STIRR also provides a natural framework for effectively visualizing the underlying relational data; this visualizer is depicted in Figure 2.

In Sections 4 and 5, we evaluate the performance of STIRR on data from synthetic as well as from real-world sources. In Section 4 we report on experience with the STIRR system on a class of inputs we will call *quasi-random inputs*. Such inputs consist of carefully "planted" structure among random noise, and are natural for testing mining algorithms. The advantage of such quasi-random data is that we may control in a precise manner the various parameters associated with the dynamical systems we use, studying in the process their efficacy at discovering the appropriate planted structure. In Section 5, we report on our experiences with STIRR on "real-world" data drawn from a range of settings. In all these settings, the main clusters computed by STIRR correspond naturally to multiple, highly correlated groups of items within the data.

2 Algorithms and basic analysis

We now describe our core algorithm, which produces a dynamical system from a table of categorical data. We begin with a table of relational data: we view such a table as consisting of a set of k *fields*, each of which can assume one of many possible *values*. (We will also refer to the fields as *columns*.) We represent each possible value in each possible field 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. See Figure 1 for an example of this representation, on a table with three fields. A *configuration* is an assignment of a weight w_v to each node v; we will refer to the entire configuration as simply w. We will 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 draw on the high-level notion of a "weight-propagation" scheme developed in the introduction.

For our purposes, a *dynamical system* is the repeated application of a function f on some set of values. A *fixed point* of a dynamical system is a point u for which $f(u) = u$. That is, it is a point which remains the same under the (repeated) application of f. Fixed points are one of the central objects of study in dynamical systems, and by iterating f one often reaches a fixed point. (See [15] for a comprehensive introduction to the mathematical study of dynamical systems.) At the end of this section, we indicate the connections between such systems and *spectral graph theory*, discussed above as a powerful method for attacking

	Attribute		
Tuple	**a**	**b**	**c**
1.	A	W	1
2.	A	X	1
3.	B	W	2
4.	B	X	2
5.	C	Y	3
6.	C	Z	3

is represented as

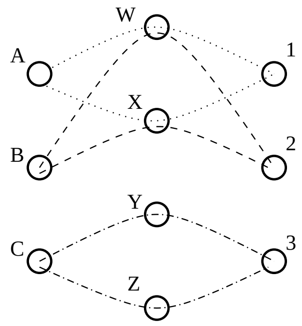

Figure 1: The representation of a collection of tuples.

discrete partitioning problems. Spectral graph theory is based on studying linear weight-propagation schemes on graph structures; much of the motivation for the weight-propagation algorithms underlying STIRR derives from this connection, though the description of our methods here will be entirely self-contained.

The dynamical system for a set of tuples will be based on a function f, which maps a current configuration to a new configuration. We define f as follows. We choose a *combiner* function $\oplus$, defined below, and update each weight w_v as follows.

To update the weight w_v:
  For each tuple $\tau = \{v, u_1, \ldots, u_{k-1}\}$
  containing v do
    $x_\tau \leftarrow \oplus(u_1, \ldots, u_{k-1})$.
  $w_v \leftarrow \sum_\tau x_\tau$.

Thus, the weight of v is updated by applying $\oplus$ separately to the members of all tuples that contain v, and adding the results. The function f is then computed by updating the weight of each w_v as above, and then normalizing the set of weights using $N()$. This yields a new configuration $f(w)$.

Iterating f defines a dynamical system on the set of configurations. We run the system through a specified number of iterations, returning the final set of configuration that we obtain. It is very difficult to rigorously analyze the limiting properties of the weight sets in the fully general case; below we discuss some analytical results that we have obtained in this direction. However, we have observed experimentally that the weight sets generally converge to fixed points or to cycles through a finite set of values. We call such a final configuration a *basin*; the term is meant to capture the intuitive notion of a configuration that "attracts" a large number of starting configurations.

To complete the description of our algorithms, we discuss the following issues: our choice of combining operators; our method for choosing an initial configuration; and some additional techniques that allow us to "focus" the dynamical system on certain parts of the set of tuples.

Choosing a combining operator. Our potential choices for $\oplus$ are the following; in Section 4 we offer some additional insights on the pros and cons among these choices.

- The product operator Π: $\oplus(w_1, \ldots, w_k) = w_1 w_2 \cdots w_k$.

- The addition operator: $\oplus(w_1, \ldots, w_k) = w_1 + w_2 + \cdots + w_k$.
- A generalization of the addition operator that we call the S_p *combining rule*, where p is an odd natural number. $S_p(w_1, \ldots, w_k) = (w_1^p + \cdots + w_k^p)^{(1/p)}$. Note that addition is simply the S_1 rule.
- A "limiting" version of the S_p rules, which we refer to as S_∞. $S_\infty(w_1, \ldots, w_k)$ is defined to be equal to w_i, where w_i has the largest absolute value among the weights in $\{w_1, \ldots, w_k\}$. (Ties can be broken in a number of arbitrary ways.)

Thus, the S_1 combining rule is *linear* with respect to the tuples. The Π and S_p rules for $p > 1$, on the other hand, involve a *non-linear* term for each individual tuple, and thus have the potential to encode co-occurrence within tuples more strongly. S_∞ is an especially appealing rule in this sense, since it is non-linear, particularly fast to compute, and also appears to have certain useful "sum-like" properties.

Non-principal basins. Much of the power of spectral graph partitioning for discrete clustering problems derives from its use of *non-principal eigenvectors.* In particular, it provides a means for assigning positive and negative weights to nodes of a graph, in such a way that the nodes with positive weight are typically well-separated from the nodes of negative weight. We will discuss this further at the end of the section.

One of our contributions here is the generalization of the notion of non-principal eigenvectors (from linear maps) to our nonlinear dynamical systems. To find the eigenvectors of a linear system $Ax = \lambda x$, the *power iteration method* maintains a set of orthonormal vectors $x^{\langle j \rangle}$, on which the following iteration is performed: First each of the $\{x^{\langle j \rangle}\}$ is multiplied by A; then the set $\{x^{\langle j \rangle}\}$ is restored to be orthormal. In an analogous way, we can maintain *several* configurations $w^{\langle 1 \rangle}, \ldots, w^{\langle m \rangle}$ and perform the following iteration:

> Update $w^{\langle i \rangle} \longleftarrow f(w^{\langle i \rangle})$, for $i = 1, 2, \ldots, m$.
> Update the set of vectors $\{w^{\langle 1 \rangle}, \ldots, w^{\langle m \rangle}\}$ so that it is orthonormal.

Assuming that a standard method is used to keep the set $\{w^{\langle i \rangle}\}$ orthornormal (see e.g. the Gram-Schmidt procedure in [22]), the configuration $w^{\langle 1 \rangle}$ iterates to a basin as before; we refer to this as the "principal basin." The updating of the other configurations is interleaved with orthonormalization steps, and we refer to these as "non-principal basins."

In the experiments that follow, we will see that these non-principal basins provide structural information about the data in a fashion analogous to the role played by non-principal eigenvectors in the context of spectral partitioning. Specifically, forcing the configurations $\{w^{\langle j \rangle}\}$ to remain orthonormal as vectors introduces *negative* weights into the configurations. Just as the nodes of large weight in the basin tend to represent natural "groupings" of the data, the positive and negative weights in the non-principal basins tend to represent good partitions of the data. The nodes with large positive weight, and the nodes with large negative weight, tend to represent "dense regions" in the data with few connections between them.

Choosing an initial configuration. There are a number of methods for choosing the initial configuration for the iteration. If we are not trying to "focus" the weight in a particular portion of the set of tuples, we can choose the *uniform initialization* (all weights set to 1, then normalized) or the *random initialization* (each weight set to an independently chosen random value in $[0, 1]$, then normalized).

For combining operators which are sensitive to the choice of initial configuration — the product rule is a basic example — one can focus the weight on a particular portion of the set of tuples by initializing a configuration *over* a particular node. To *initialize* w *over the node* v, we set $w_u = 1$ for every node u appearing in a tuple with v, and $w_{u'} = 0$ for every other node u'. (We then normalize w.) In this way, we hope to cause nodes "close" to v to acquire large weight. This notion will be addressed in the experiments of the following sections.

Masking and modification. To augment or diminish the influence of certain nodes, we can apply a range of "local modifications" to the function f. Specifically, for a particular node v, we can compose f with the operation $mask(v)$, which sets $w_v = 0$ at the end of each iteration. Alternately, we can compose f with $augment(v, x)$, which adds a weight of $x > 0$ to w_v at the end of each iteration. This latter operation "favors" v during the iterations, which can have the effect of increasing the weights of all nodes that have significant co-occurrence in tuples with v.

Relation to other Iterative Techniques

The body of mathematical work that most closely motivates our current approach is *spectral graph theory*, which studies certain algebraic invariants of graphs. We motivate this technical background with the following example.

Example. Consider a triangle: a graph G consisting of three mutually connected nodes. At each node of G we place a positive real number, and then we define a dynamical system on G by iterating the following map: the number at each node is updated to be the *average* of the numbers at the two other nodes. The analysis of this dynamical system is not difficult: the numbers at the three nodes converge to one another as the map is iterated.

Spectral graph theory is based on the relation between such iterated linear maps on an undirected graph G and the eigenvectors of the *adjacency matrix* of G. (If G has n nodes, then its adjacency matrix $A(G)$ is a symmetric n-by-n matrix whose (i, j) entry is 1 if node i is connected to node j in G, and 0 otherwise.) The principal eigenvector of A can be shown to capture the equilibrium state of a linear dynamical system of the form in the above example; the non-principal eigenvectors provide information about good "partitions" of the graph G into dense pieces with few interconnections. The most basic application of spectral graph partitioning, which motivates some of our algorithms, is the following. Each non-principal eigenvector of the adjacency matrix of G can be viewed as a labeling of the nodes of G with real numbers — some positive and some negative. Treating the nodes with positive numbers as one "cluster" in G, and the nodes with negative numbers as another, typically partitions the underlying graph into two relatively dense pieces, with few edges in between.

The iteration of systems of non-linear equations arises also in connection with neural networks [32, 33]. In many of these neural network settings, the emphasis is quite different from ours — for example, one is concerned with the approximation of binary decision rules, or with the approximation of connection strengths among nodes on a discrete lattice (e.g. the work on Kohonen maps [29]). In our work, on the other hand, the analogues of "connections" — the co-occurrences among tuples in a database — typically do not have any geometric structure that can be exploited, and our approach is to treat the connections as fixed while individual item weights are updated.

In a related vein, the methodology of *principal curves* allows for a non-linear approach to dimension-reduction, which can sometimes achieve significant gains over linear methods such as principal component analysis [24, 30]. This method thus implicitly requires an embedding in a given space in which to perform the dimension-reduction; once this dimension-reduction is performed, one can then approach the low-dimensional clustering problem on the data in a number of standard ways.

Analysis

There is much that remains unknown about the dynamical systems we use here, and this reflects the lack of existing mathematical techniques for proving properties of general non-linear dynamical systems. This of course does not prevent them from being useful in the analysis of categorical tables; and we now report on some of the basic properties that we are able to establish analytically about the dynamical systems we use. In addition, we state some basic conjectures that we hope will motivate further work on the theoretical underpinnings of these models.

We first state a basic convergence result for the (linear) S_1 combining rule, subject to a standard non-degeneracy assumption in linear algebra: that the set of eigenvalues associated with the linear update rule has a unique maximum element. (We do not go into further details here on what happens when this assumption does not hold; this is fairly well understood in the literature).

Theorem 1 *For every set T of tuples, and every initial configuration w, w converges to a fixed point under the repeated application of the combining rule S_1.*

The proof is based on relating the fixed points of this system to the eigenvectors of an undirected graph derived from the set T of tuples.

If we consider iterating a linear system on an undirected graph G (as in the power iteration method discussed above), there is a natural *combinatorial* interpretation of the weights that are computed: they count the number of walks in G starting from each vertex. We now show that the product rule Π has an analogous combinatorial interpretation for a set T of tuples. This combinatorial interpretation indicates a precise sense in which the product rule discovers "dense" regions in the data.

To state this result, we need some additional terminology. If T is a set of tuples over a dataset with d fields, v is a node of T, and k is a natural number, then a (T, v, k)-*tree* is a complete $(d-1)$-ary tree Z of height k whose vertices are labeled with nodes of T as follows. (1) The root is labeled with v. (2) Suppose a vertex α of Z is labeled with a node $u \in T$; then there is a tuple τ of T so that $u \in \tau$ and the children of α are labeled with the elements of $\tau - \{u\}$. Two (T, v, k)-trees are *equivalent* if there is a one-to-one correspondence between their vertices that respects the tree structure and the labeling; they are *distinct* otherwise.

Note that if G is a graph, then a (G, v, k)-tree is precisely a walk of length k in G starting from v.

Theorem 2 *Let T be a set of tuples, and consider the dynamical system defined on it by the product rule Π. For a node v of T, let w_v^k denote the weight w_v assigned to v after k iterations. Then w_v^k is proportional to the number of distinct (T, v, k)-trees.*

Next, we establish a basic theorem that partially underlies our intuition about the partioning properties of these dynamical systems. We state it for the product rule, but analogous results can be shown for the other combining rules we consider. By a *complete hypergraph of size s*, we mean a set of tuples over a dataset with s nodes in each column, so that there is a tuple for every choice of a node from each column. Let T_{ab} denote a set of tuples consisting of the disjoint union of a complete hypergraph A of size a and a complete hypergraph B of size b. We define 1_A to be the configuration that assigns a weight of 1 to each node in A, and a weight of 0 to each node in B; we define 1_B analogously.

Theorem 3 *(i) Let w be a randomly initialized configuration on the set of tuples T_{ab}, $a > b$. Then with probability at least $1 - \exp(b - a)$, w will converge under the product rule Π to the fixed point $N(1_A)$.*

(ii) Let w' be a configuration initialized over *a random node $v \in H_{ab}$. Then with probability $\frac{a}{a+b}$, w' converges to $N(1_A)$ under Π, and with probability $\frac{b}{a+b}$ it converges to $N(1_B)$.*

We conjecture that qualitatively similar behavior occurs when T_{ab} consists of two "dense" sets of tuples that are "sparsely" connected to each other. At an experimental level, this is borne out by the results of subsequent sections.

The previous theorem shows that it is possible for the product rule to converge to different basins, depending on the starting point. We now formulate a general statement that addresses a notion of *completeness* for such systems: all sufficiently "large" basins should be discovered within a reasonable amount of time. Let us say that a basin w^* has *measure ϵ* if at least an ϵ fraction of all random starting configurations converge to w^* on iterating a given combining rule. Let B_ϵ be the number of such basins.

Theorem 4 *With probability at least $1 - 1/B_\epsilon$, iterating the system from $2\epsilon^{-1} \ln B_\epsilon$ random initial configurations is sufficient to discover all basins of measure at least ϵ.*

This gives a precise sense — via a standard type of sampling result — in which all sufficiently large basins will be discovered rapidly by iterating from random starts. (Note that this discovery is relatively efficient — since B_ϵ can be as large as ϵ^{-1}, it can take up to ϵ^{-1} starting points to discover all the basins, and our randomized approach uses $2\epsilon^{-1} \ln B_\epsilon$.)

3 Overview of Experiments

3.1 System overview

The Stirr experimental system consists of a computation kernel written in C, and a control layer and GUI written in Tcl/Tk. We prepared input datasets in straightforward whitespace-delimited text files. The output can be viewed in a text window showing nodes of highest

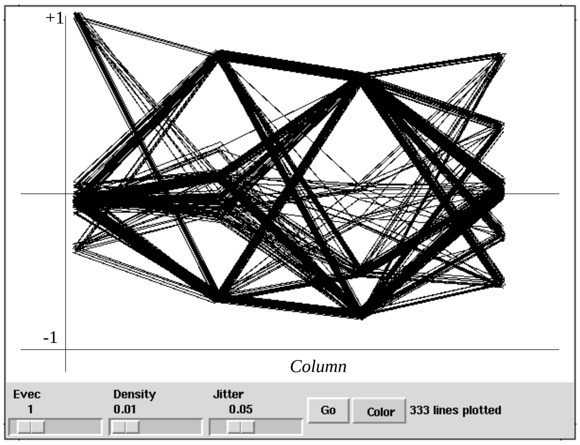

Figure 2: Graphical view of system

weight in each column, or as a graphical plot of node weights. The GUI allows us to modify parameters quickly and easily.

Figure 2 depicts a graphical view of the first non-principal basin $w^{\langle 2 \rangle}$ for an adult census dataset, using just 4 of the categorical attributes. Each node is positioned according to its column and calculated weight. The nodes in each tuple are joined by a line (consisting of three segments in this case). The *Density* control indicates only 1% of the tuples were plotted. The *Jitter* value "smears" each line's position by a small random amount, to separate lines meeting at the same point. There is a simple tuple coloring mechanism, which changes the color of a selected line to show the weight assignment on that tuple.

The value of the visualizer in presenting similar nodes to the user is evident in Figure 2; this may be used, for instance, to point to a subset of nodes for masking or modification, as discussed earlier. It could thus be a useful component in an interactive data mining tool.

3.2 Performance

Our algorithm scales well with increasing data volumes. We measured the running time for 10 iterations, for varying numbers of tuples and numbers of columns. The S_∞ combiner was used, and 4 non-principal basins were computed. The plot in Figure 3.2 demonstrates that the running time is linear in number of tuples, and nearly linear in number of columns — the slight variation is due to varying numbers of nodes per column. Qualitatively similar plots of running times are obtained for the other main combining rules.

The running times were measured on a 128MB, 200 MHz Pentium Pro machine, running Linux, with no other user processes. The dataset was generated randomly.

4 Quasi-random inputs

In this section we report on experience with the STIRR system on the class of *quasi-random inputs* discussed in the introduction. For the purposes of testing the performance of a mining algorithm, such inputs are a natural setting in which to search for "planted" structures

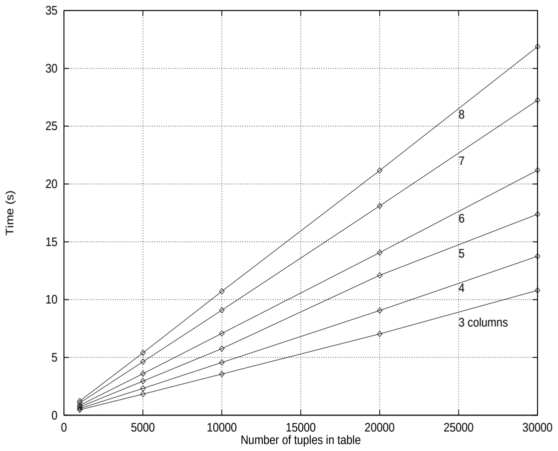

Figure 3: Running times. We have truncated the y axis for compactness; all the lines remain straight up to 1000000 tuples.

among random noise. This is a technique used in the study of computationally difficult problems. For instance, in combinatorial optimization, the work of Boppana [7] analyzes graph bisection heuristics in quasi-random graphs in which a "small" bisection is hidden; Blum and Spencer [6] adopt the same approach to the analysis of graph-coloring algorithms. The advantage of such quasi-random data is that we may control in a precise manner the various parameters associated with the dynamical systems we use, studying in the process their efficacy at discovering the appropriate planted structure. Consider 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 may be thought of as a "cluster" in the data, since these nodes co-occur more often in the table than in a purely random table. (The analog in numerical data with random multidimensional points is a region with a higher probability density than the ambient.) Our hope, then, might be that STIRR discovers such clusters.

We now discuss the aspects of STIRR that we study through this approach; we describe the quasi-random experiment in each case.

(1) How well does STIRR distil a cluster in which the nodes have an above-average rate of co-occurrence? Intuitively, this is of interest because its analog in real data could be the set of people (in a travel database) with similar travel patterns, or a set of failures (in a maintenance database) that stem from certain suppliers, manufacturing plants, etc. Whereas a conventional data analysis technique would be faced with enumerating all cross-products across columns of all possible subsets in each column, and examining the support of each, the STIRR technique appears to avoid this prohibitive computation. How quickly does STIRR elicit such structure (say, as a function of the number of iterations, the relative density of the cluster to the background, the combine operator $\oplus$, etc.)? Our experiments show that for quasi-random inputs, such structure emerges very quickly, typically in 5 to 10 iterations. Figure 4 depicts this experiment.

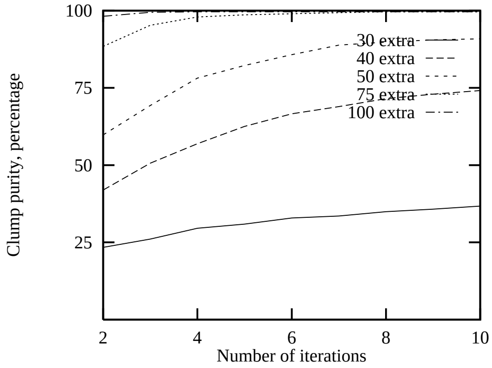

Figure 4: Finding planted data. Table has 3 columns, 1000 distinct attributes per column, 5000 rows; 'extra' rows are planted tuples, using attributes from a set of 30 (10 per column); 'purity' is the percentage of these attributes ending up in the top 10 positions of the principal basin. $\oplus$ is S_∞.

(2) How well does Stirr separate distinct planted clusters using non-principal basins? For instance, if in a random table we were to plant two clusters each involving a distinct set of nodes, will the first non-principal basin place the clusters at opposite ends of the partition? Again, this is a basic task in analyzing large categorical tables — discovering well-separated sub-populations — and we hope to avoid the exponential cost of NP-hard formulations of this as combinatorial graph decomposition problems. Intuition from spectral partitioning suggests that typically some non-principal partition will separate a pair of clusters; our experiments show that Stirr usually achieves such a separation within the first non-principal basin, for quasi-random inputs. Figure 5 summarizes this experiment, showing that the separation achieved stabilizes after about 15 iterations. We make the stringent demand in this experiment that the separation manifest itself in the first non-principal basin (rather than allowing any non-principal basin). Let a_0 and b_0 be the numbers of nodes from clusters A and B at one end of this basin, and a_1 and b_1 be the corresponding numbers at the other end. Under perfect separation we would have $a_0 = 30$ and $b_1 = 30$, or vice-versa. Our separation measure in this case (it generalizes easily to other cluster and table sizes) is

$$s(A, B) = \frac{|a_0 - b_0| + |a_1 - b_1|}{60}.$$

(3) How well does Stirr cope with tables containing clusters in a few columns, with the remaining columns being random? This is standard problem in data mining, where we seek to mask out irrelevant factors (here columns) from a pattern. We study this by adding (to a randomly generated table) additional random tuples concentrated on a few nodes in 3 of the columns; the entries for the remaining columns are randomly distributed over *all* nodes. Thus these remaining columns correspond to irrelevant attributes that might (by their totally random distribution) mask the cluster planted in columns 1–3. Our experiments show that

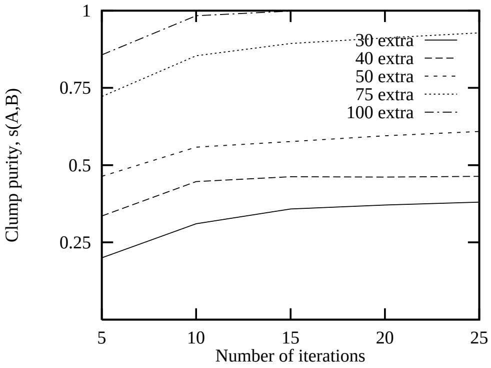

Figure 5: Separating two planted clusters. Extra rows now drawn from two distinct sets of 30 attributes. $s(A, B)$ is the difference in sizes of the two subsets of these attributes in the top 10 positions, summed over both ends of the first non-principal basin, and normalized.

STIRR will indeed mask out such "irrelevant" attributes; we do not know of an analog for this phenomenon in spectral graph partitioning. As Figure 6 shows, as the number of redundant columns is increased, STIRR suffers a relatively modest and graceful loss of purity.

(4) How does the choice of combine operator ($\oplus$) affect speed of convergence? Quasi-random experiments provide a controlled way to investigate this; figure 7 shows experiments to assess convergence rate for different combine operators. The S_∞ (Max) operator is the clear winner. The product rule Π did not converge monotonically, so one may consider second place in this experiment to belong to the S_1 rule; or to S_2 or S_5 if the arithmetic speed can be improved.

Each data point in each experiment is an average over 100 runs: using 10 random tables, with 10 random starting points. The the number of entities in the table here is fixed; we obtained similar results for other table sizes. The convergence requires very few iterations, even with the fairly strict separation measure used in the two-cluster experiment.

5 Real data

We now discuss some of our experience with this method on a variety of "real" data sets.

Basic data sets. First, we consider two data sets that illustrate co-occurrence patterns.

(1) *Bibliographic data.* We took publicly-accessible bibliographic databases of 7000 papers from database research [38], and of 30,000 papers written on theoretical computer science and related fields [34], and constructed two data sets each with four columns as follows. For each paper, we recorded the name of the first author, the name of the second author, 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).* Note that

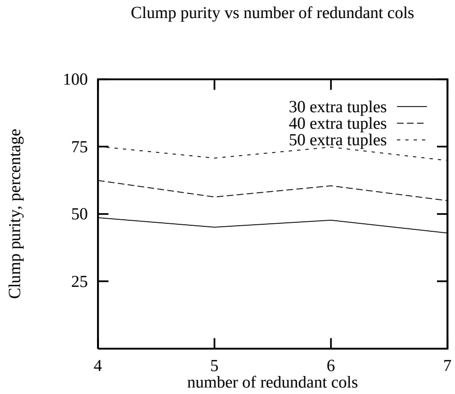

Figure 6: Adding redundant columns. We augment the tables of figure 4 with columns of random attributes, and examine purity after 10 iterations.

we deliberately treat the year, although a numerical field, as a categorical attribute for the purposes of our experiments. In addition to the tables of theory papers and database papers, we also constructed a mixed table with both sets of papers.

(2) *Login Data.* From four heavily used server hosts within IBM Almaden, we compiled the history of user logins over a several-month period. This represented a total of 47,000 individual logins. For each login, we recorded the user name, the remote host from which the login occurred, the hour of the login, and the hour of the corresponding logout. Thus we created the tuples of the form: *(user, remote-host, login-time, logout-time).* Again, the hours were treated as categorical attributes.

Sequential Data Sets. We also investigated a natural approach by which this method can be used to analyze sequential datasets. The hope here is to use local co-occurrences in time to elicit local cause-effect patterns in the data. We approach this as follows. Given a long string of sequentially occurring events, we label them as $e_1, e_2, \ldots, e_n$. We then imagine a "sliding window" of length ℓ passing over the data, and construct ℓ-tuples $(e_i, e_{i+1}, \ldots, e_{i+\ell-1})$ for each i between 1 and $n - \ell + 1$. For data exhibiting a strong local cause-effect structure, we hope to see such structure emerge from the co-occurrences among events close in time.

This framework bears some similarity to a method of Mannila, Toivonen, and Verkamo [31] for defining frequent "episodes" in sequential data. Our approach is different from theirs, however, since we do not search separately for each discrete type of episode; rather, we use the core algorithm of STIRR to group events that co-occur frequently in temporal proximity, without attempting to impose an episode structure on them.

One example we studied was drawn from a readily accessible source of sequential data, exhibiting strong elements of causality: chess moves. Often, the identity of a single move in a chess game allows one to make plausible inferences about the identity of the moves that occurred nearby in time. We worked with 4-move windows, on a dataset of moves from 2000 chess games. Many of the basins discovered by STIRR represented common local motifs in the opening and middle-game; due to lack of space, we do not discuss this setting further.

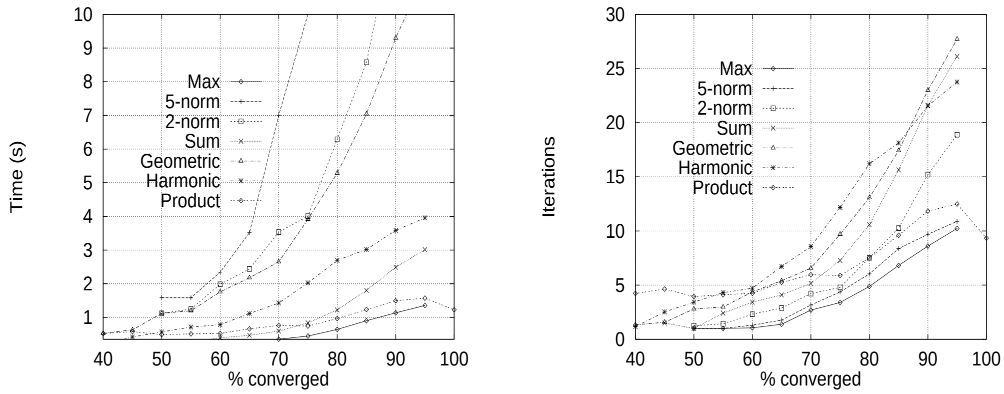

Figure 7: Convergence for different $\oplus$. Table has 2100 rows: 1000 using one set of attributes, A, and 1100 with another, B. Convergence is measured as the number of B attributes in the top 20 positions. The experiment measured average running times and iterations until 90% convergence.

Overview. At a very high level, the following phenomena emerge. First, the principal and non-principal basins computed by the method correspond to dense regions of the data with natural interpretations. For the sequential data, the nodes with large weight in the main basins exhibit a natural type of cause-effect relationship in the domains studied.

The fact that we have treated numerical attributes as categorical data — without attempting to take advantage of their numerical values — allows us to make some interesting observations about the power of the method at "pulling together" correlated nodes. Specifically, we will see in the examples below that nodes which are close as numbers are typically grouped together in the basin computations, *purely through their co-occurrence with other categorical attributes.* This suggests a strong sense in which co-occurrence among categorical data can play a role similar to that of proximity in numerical data, for the purpose of clustering. We divide the discussion below into two main portions, based on the combining rule $\oplus$ that we use.

5.1 The S_p Rules

We now discuss our experience with the S_p combining rules. We ran the S_∞ dynamical system on a table derived as described above from a mixture of theory and database research papers. The first non-principal basin point is quite striking: in Figure 8 we list the ten items with the most positive and most negative weights in each column (together with their weights); a double horizontal line separates these groups. (Note that no row in Figure 8 need be a tuple in the input; the table simply lists, for each column, the ten nodes with the most positive (and the ten with the most negative) values.) Note also that our parser identified all people by their last names so that, for instance, the Chen at the top is actually several Chens taken together. A number of phenomena are apparent from the above summary views of the theory and database communities (the fact that the theory community ended up at

0.1811: Chen	0.1629: Chen	0.317: TCS	0.563: 1995
0.1185: Chang	0.1289: Wang	0.2264: IPL	0.5596: 1994
0.1147: Zhang	0.1007: COMPREVS	0.195: LNCS	0.4332: 1996
0.1119: Agarwal	0.1: Li	0.1633: INFCTRL	0.151: 1993
0.1104: Bellare	0.09004: Rozenberg	0.1626: DAMATH	0.07487: 1992
0.1053: Gu	0.08837: Igarashi	0.1464: JPDC	0.0155: 1976
0.09467: Dolev	0.08171: Sharir	0.1371: SODA	0.005276: 1972
0.08571: Hemaspaand	0.0797: Huang	0.1266: STOC	0.004569: 1985
0.08423: Farach	0.07924: Maurer	0.1074: IEEETC	0.001891: 1973
0.08232: Ehrig	0.0783: Lee	0.1002: JCSS	0.001679: 1970
-0.1195: Stonebrake	-0.1068: Wiederhold	-0.384: IEEEDataEng	-0.2165: 1986
-0.1059: Agrawal	-0.09615: David	-0.256: VLDB	-0.1983: 1987
-0.1023: Wiederhold	-0.08343: DeWitt	-0.2392: SIGMOD	-0.1519: 1989
-0.07735: Abiteboul	-0.07643: Richard	-0.1504: PODS	-0.14: 1988
-0.06783: Yu	-0.07454: Stonebrak	-0.1397: ACMTDS	-0.11: 1984
-0.06722: Navathe	-0.07403: Michael	-0.1176: IEEETransa	-0.06571: 1975
-0.06615: Litwin	-0.07159: Robert	-0.04832: IEEETrans	-0.06431: 1990
-0.06308: Bernstein	-0.06843: Jagadish	-0.04658: IEEETechn	-0.03765: 1980
-0.0623: Jajodia	-0.06722: James	-0.04533: WorkshopI	-0.03238: 1974
-0.0587: Motro	-0.0671: Navathe	-0.03581: IEEEDBEng	-0.02873: 1982

Figure 8: Separating a mixture of theory and database papers

the positive end of the partition and the database community at the negative end is not significant – this depends only on the random initial configuration, and could just as well have been reversed). What is significant is that such a clean dissection resulted from the dynamical system, and that too within the first (rather than some lower) non-principal basin. Note that these phenomena have been preserved despite some noise in the database research bibliography: first names on singly-authored papers have sometimes been parsed as second authors.

5.2 The Product Rule

The high-level observations above apply to our experience with the combining rule Π; but the sensitivity of Π to the initial configuration leads to some additional recurring phenomena.

(1) First, although there is not a unique fixed point that all initial configurations converge to, there is generally a relatively small number of large basins. Thus we can gain additional "similarity" information about initial configurations by looking at those configurations that reach a common basin.

(2) The short-term temporal behavior of the dynamical system defined by Π, as we go through the first few iterations, can be quite informative. In particular, as a configuration moves towards its ultimate basin, the configurations through which it passes can provide interesting information about groupings of certain nodes Thus, our method is capable of providing structural information about the data both through the *dynamics* of the process, as well as through the static basin results.

(3) The "masking" of frequently occurring nodes can be a useful way to uncover hidden structure in the data. In particular, a frequent node can force nearly all initial configurations to a common basin in which it receives large weight. By masking this node, one can discover a large number of additional basins that otherwise would have been "overwhelmed" in the iterations of the dynamical system.

Bibliographic Data. We begin with the bibliographic data on theoretical CS papers. There are a number of co-occurrences that one might hope to analyze in this data, involving authors, conferences, and dates. Thus, if we choose an initial configuration centered around the name of a particular conference and then iterate the product rule, we generally obtain a basin that corresponds naturally to the "community" of individuals who frequently contribute to this conference, together with similar conferences.

Initializing a configuration over a specific year leads to another very interesting phenomenon, alluded to above: the basin groups "nearby" years together, purely based on co-occurrences in the data. It is of course natural to see how this should happen; but it does show a concrete way in which co-occurrence mirrors more traditional notions of "proximity." For example, initializing over the year 1976 leads to the basin in Figure 9. Observe that the years grouped with 1976 are 1978, 1977, 1975, and 1985. We observe also that this is a fairly descriptive summary view of theoretical CS research around 1976. It is important to note that initializing over different years can lead to the same basin; this, as discussed above, is due to the "attracting" nature of strong basins. Thus, for example, we obtain exactly the same basin when we initialize over 1974, although 1974 does not ultimately show up among the top 5 years when the basin is reached.

Login Data. We now discuss some examples from the login data introduced above. For this data, there was one user who logged in/out very frequently, gathering large weight in our dynamical system, almost regardless of the initial configuration. Thus for all the random initializations tried on this data set, a common basin was reached.

In order to discover additional structure, it proved very useful to *mask* this user, via the *masking* operation introduced above. Users in this data can exhibit similarities based on co-occurrence in login times and machine use. If we first mask the extremely common user(s), and then initialize the configuration over the user `root`, such further structure emerges: the four highest-weight users turn out to be `root`; a special "help" login used by the system administrators; and the individual login names of two of the system administrators. Thus STIRR discerns system administrators based on their co-occurrences across columns with `root`.

Finally, we illustrate another case in which "similar" numbers are grouped together by

0.697 Garey	.824 Johnson	.344 JACM	.461 1976
0.241 Aho	.059 Hirschberg	.228 SICOMP	.152 1978
0.026 Yu	.039 Hopcroft	.123 TCS	.122 1977
0.018 Hassin	.030 Graham	.123 TRAUB	.095 1975
0.004 Chandra	.009 Tarjan	.046 IPL	.082 1985

Figure 9: A cluster of related years

co-occurrence, rather than by numerical proximity. When we initialize over the hour 23 (i.e., 11pm) as the login time, we obtain a set of users with predominantly late login times, and we obtain the following hours as the highest-weight nodes for login and logout times:

0.430	22	.457	22
0.268	21	.286	23
0.173	23	.134	21
0.079	20	.060	00

Thus the algorithm concludes that the hours $22, 21, 23, 20$, and 00 are all "similar," purely based on the login patterns of users.

6 Conclusions and Further Work

We have given a new method for clustering and mining tables of categorical data, by representing them as non-linear dynamical systems. The two principal contributions of this paper are: (1) we generalize ideas from spectral graph analysis to hypergraphs, bridging the power of these techniques to clustering categorical data; (2) we develop a novel connection between tables of categorical data and nonlinear dynamical systems. Experiments with the STIRR system show that such systems can be effective at uncovering similarities and dense "sub-populations" in a variety of types of data; this is illustrated both through explicitly constructed "hidden populations" in our quasi-random data, as well as on real data drawn from a variety of domains. Moreover, these systems converge very rapidly in practice, and hence the overall computation effort is typically linear in the size of the data.

The connection between dynamical systems and the mining of categorical data suggests a range of interesting further questions. At the most basic level, we are interested in determining other data mining domains in which this approach can be effective. We also feel that a more general analysis of the dynamical systems used in these algorithms — a challenging prospect — would be interesting at a theoretical level and would shed insight into further techniques of value in the context of mining categorical data.

Acknowledgements

We thank Rakesh Agrawal for many valuable discussions on all aspects of this research, and for his great help in the presentation of the paper. We also thank Bill Cody and C. Mohan for their valuable comments on drafts of this paper. The work of the first and second authors was performed in part while visiting the IBM Almaden Research Center. The second author is supported in part by an Alfred P. Sloan Research Fellowship and by NSF Faculty Early Career Development Award CCR-9701399.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthu-

rusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT Press, 1996.

[2] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. *Proc. SIGMOD*, 1993.

[3] A. Agresti. Categorical Data Analysis. Wiley-Interscience, 1990.

[4] N. Alon, "Eigenvalues and expanders," *Combinatorica*, 1986.

[5] C. Berge, *Hypergraphs*, North-Holland, 1973.

[6] A. Blum, J. Spencer, "Coloring random and semi-random k-colorable graphs." *J. Algorithms*, 19(1995).

[7] R. Boppana, "Eigenvalues and graph bisection: An average-case analysis," *Proc. IEEE Symp. on Foundations of Computer Science*, 1987.

[8] S. Brin, R. Motwani, J.D. Ullman, S. Tsur. "Dynamic itemset counting and implication rules for market basket data". *Proc. ACM SIGMOD*, 1997.

[9] S. Brin, L. Page. "Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. 7th International World Wide Web Conference*, 1998.

[10] E. Charniak, *Statistical Language Learning*, MIT Press, 1993.

[11] T. Chiueh, "Content-based image indexing," *Proc. VLDB Conference*, 1994.

[12] F.R.K. Chung, *Spectral Graph Theory*, AMS Press, 1997.

[13] G. Das, H. Mannila, P. Ronkainen, "Simliarity of attributes by external probes," manuscript, 1997.

[14] S. Deerwester, S. T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. Journal of the Society for Information Science, 41(6), 391-407, 1990.

[15] R.L. Devaney, *An Introduction to Chaotic Dynamical Systems*, Benjamin Cummings, 1989.

[16] W.E. Donath, A.J. Hoffman, "Lower bounds for the partitioning of graphs", *IBM Journal of Research and Development*, 17(1973). 420–425.

[17] R.O. Duda and P.E. Hart. Pattern Classification and Scene Analysis. Wiley, New York, 1973.

[18] C. Faloutsos. Searching Multimedia Databases by Content. Kluwer Academic, 1996.

[19] M. Fiedler, "Algebraic connectivity of graphs," *Czech. Math. Journal* 23(1973), pp. 298–305.

[20] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker. Query by image and video content: The QBIC system. IEEE Computer, **28**, 23–32, 1995.

[21] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

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

[23] Eui-Hong Han, George Karypis, Vipin Kumar, Bamshad Mobasher, "Clustering Based On Association Rule Hypergraphs," *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[24] T. Hastie, W. Stuetzle, "Principal curves," *J. Am. Stat. Assoc.* 84(1989), pp. 502–516.

[25] Zhexue Huang, "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining." *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[26] M. Jerrum, A. Sinclair, "The Markov chain Monte Carlo method: An approach to approximate counting and integration," in *Approximation Algorithms for NP-hard Problems*, D.S.Hochbaum ed., PWS Publishing, 1996

[27] I.T. Jolliffe. Principal Component Analysis. Springer-Verlag, 1986.

[28] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1998. Also available as IBM Research Report RJ 10076(91892) May 1997.

[29] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Bio. Cybern.*, 43(1982), pp. 59-69.

[30] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal* 37(1991), pp. 233-243.

[31] H. Mannila, H. Toivonen, A.I. Verkamo, "Discovering frequent episodes in sequences," *Proc. KDD Conf.*, 1995.

[32] H. Ritter, T. Martinetz, *Neural Computation and Self-Organizing Maps*, Addison-Wesley, 1992.

[33] D.E. Rumelhart, J.L. McClelland, eds., *Parallel and Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.

[34] J. Seiferas, *A large bibliography on theory/foundations of computer science*, at `http://liinwww.ira.uka.de/bibliography/Theory/Seiferas/`.

[35] M. Shub, personal communication.

[36] D. Spielman, S. Teng, "Spectral partitioning works: Planar graphs and finite-element meshes," *Proc. IEEE Symp. on Foundations of Computer Science*, 1996.

[37] H. Toivonen. Sampling large databases for finding association rules. *Proc. VLDB Conference*, 1996.

[38] G. Wiederhold, *Bibliography on database systems*, at `http://liinwww.ira.uka.de/bibliography/Database/Wiederhold/`.

[39] T. Zhang, R. Ramakrishnan, M. Livny. Birch: An efficient data clustering method for very large databases. *Proc. ACM SIGMOD*, 1996.